МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

ВВЕДЕНИЕ В СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

Часть I

Учебно-методическое пособие

Составители:
А. В. Тучин, Е. Н. Бормонтов,
К. Г. Пономарев

Воронеж
Издательский дом ВГУ
2017

Утверждено научно-методическим советом физического факультета 1 июня 2017 г., протокол № 6

Рецензент – доктор физико-математических наук, профессор кафедры физики твердого тела и наноструктур ВГУ С. И. Курганский

Учебно-методическое пособие подготовлено на кафедре физики полупроводников и микроэлектроники физического факультета Воронежского государственного университета.

Рекомендовано для студентов 1-го курса очной формы обучения физического факультета, обучающихся по программам магистратуры.

Для направлений: 03.04.03 – Радиофизика; 11.04.04 – Электроника и наноэлектроника

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Систем автоматизированного проектирования (САПР) позволяют существенно повысить производительность труда разработчиков в различных сферах производства, за счет переноса сложных вычислений и однотипных операций на компьютер или сервер. Практически весь процесс разработки современных интегральных микросхем осуществляется средствами специализированных САПР. Знание основ работы САПР микросхем позволяет будущим разработчикам быстрее включиться в процесс разработки реальных кристаллов.

В пособии кратко описана история развития систем автоматизированного проектирования микросхем, представлен типовой маршрут проектирования цифровых микросхем, освещены вопросы физической верификации топологии, а также правила разработки цифровых ячеек и цифровых библиотек.

1. САПР: ОСНОВНЫЕ ПОНЯТИЯ И ИСТОРИЯ РАЗВИТИЯ

1.1. Введение

В настоящее время практически все конструкторские бюро и дизайн-центры оснащены компьютерами, поэтому реализация идей инженерами с помощью кульмана, карандаша и логарифмической линейки стали неактуальными. Применение программных пакетов, призванных создавать конструкторскую и технологическую документацию, 3D модели и чертежи (систем автоматизированного проектирования, САПР), конструкторами, технологами, архитекторами, исследователями, программистами стало практически повсеместным. Поэтому инженеры-разработчики должны обладать знаниями основ автоматизации проектирования и уметь работать со средствами САПР.

Из этой главы вы узнаете:

- ✓ *Что такое системы автоматизированного проектирования (САПР)*
- ✓ *Области применения САПР*
- ✓ *Цели и задачи САПР*
- ✓ *Классификацию САПР*
- ✓ *В чем отличия между САПР, CAD, CAM, CAE и EDA*
- ✓ *Историю развития САПР интегральных микросхем*
- ✓ *По каким направлениям развиваются САПР*

1.2. Понятие САПР, цели и задачи

Система автоматизированного проектирования — *автоматизированная система, реализующая информационную технологию выполнения функций проектирования, представляет собой организационно-техническую систему, предназначенную для автоматизации процесса проектирования, состоящую из персонала и комплекса технических, программных и других средств автоматизации его деятельности*. Также для обозначения подобных систем широко используется аббревиатура САПР.

Для обозначений всего спектра различных технологий автоматизации с помощью компьютера в англоязычной литературе используется термин CAx (англ. *computer-aided technologies*). Например, CAD-system (*computer-aided design*) – автоматизированное проектирование, CAE-system (*computer-aided engineering*) – системы автоматизации инже-

нерных расчетов и анализа, CAM-system (*computer-aided manufacturing*) – системы автоматизации технологических расчетов. Русскоязычное понятие САПР более общее и включает специализированные понятия англоязычной литературы.

Важно отметить, что слово "*автоматизированный*" подчёркивает участие человека в процессе, так как в САПР часть функций выполняет человек, а автоматическими являются только отдельные проектные операции и процедуры.

Драйвером развития САПР стали научно-исследовательские предприятия военно-промышленных комплексов. В США первый аппаратно-программный комплекс для систем противовоздушной обороны был создан в 1947 г. Первая советская САПР была разработана в конце 1980-х гг.

На сегодняшний момент, в большинстве случаев САПР решает задачи автоматизации работ на стадиях проектирования и подготовки производства.

Основная цель создания САПР — повышение эффективности труда инженеров, включая:

- сокращение трудоёмкости проектирования и планирования;
- сокращение сроков проектирования;
- сокращение себестоимости проектирования и изготовления, уменьшение затрат на эксплуатацию;
- повышение качества и технико-экономического уровня результатов проектирования;
- сокращение затрат на натурное моделирование и испытания.

Достижение этих целей обеспечивается путём:

- автоматизации оформления документации;
- информационной поддержки и автоматизации процесса принятия решений;
- использования технологий параллельного проектирования;
- унификации проектных решений и процессов проектирования;
- повторного использования проектных решений, данных и наработок;
- стратегического проектирования;
- замены натурных испытаний и макетирования математическим моделированием;
- повышения качества управления проектированием;
- применения методов вариантного проектирования и оптимизации.

Именно поэтому предприятия, работающие без САПР или использующие ее в малой степени, становятся неконкурентоспособными, а области применения САПР постоянно расширяются и охватывают сферы энергетики, медицины, микроэлектроники, автопромышленность, дизайн, строительство, научные исследования и многое другое (рис. 1.1).

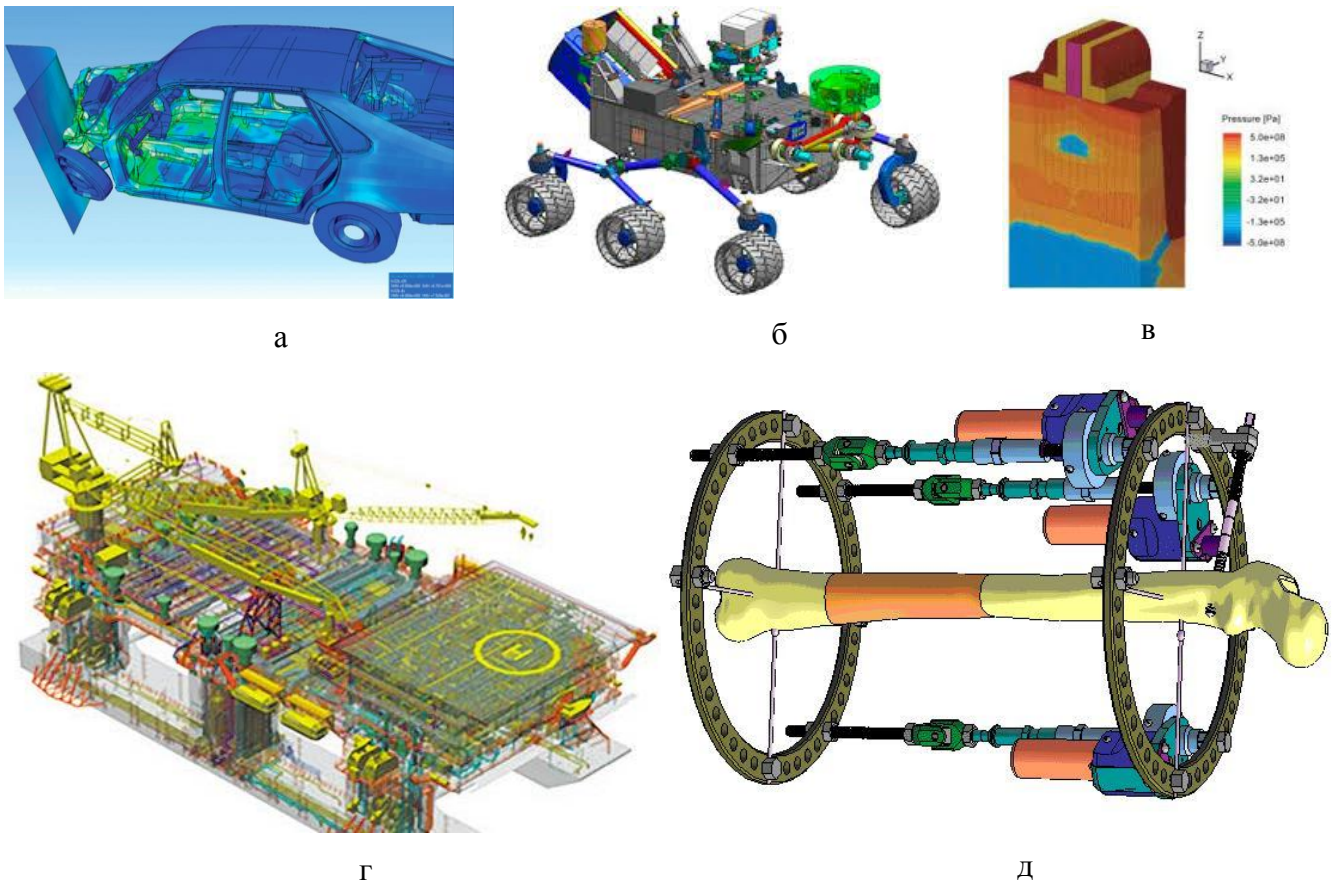


Рис. 1.1. Области применения САПР: а – автомобильная промышленность; б – космос; в – микроэлектроника; г – нефтедобыча; д – медицина

1.3. Обеспечение САПР

Выделяют следующие виды обеспечения САПР:

Техническое обеспечение — совокупность связанных и взаимодействующих технических средств (ЭВМ, периферийные устройства, сетевое оборудование, линии связи, измерительные средства).

Математическое обеспечение, объединяющее математические методы, модели и алгоритмы, используемые для решения задач автоматизированного проектирования.

Программное обеспечение, которое подразделяется на общесистемное и прикладное:

прикладное ПО реализует математическое обеспечение для непосредственного выполнения проектных процедур. Включает пакеты прикладных программ, предназначенные для обслуживания определенных этапов проектирования или решения групп однотипных задач внутри различных этапов;

общесистемное ПО предназначено для управления компонентами технического обеспечения и обеспечения функционирования прикладных программ. Примером компонента общесистемного ПО является операционная система.

Информационное обеспечение — совокупность сведений, необходимых для выполнения проектирования. Состоит из описания стандартных проектных процедур, типовых проектных решений, комплектующих изделий и их моделей, правил и норм проектирования. Основная часть информационного обеспечения САПР — базы данных.

Лингвистическое обеспечение — совокупность языков, используемых в САПР для представления информации о проектируемых объектах, процессе и средствах проектирования, а также для осуществления диалога "проектировщик – ЭВМ" и обмена данными между техническими средствами САПР. Включает термины, определения, правила формализации естественного языка. В лингвистическом обеспечении выделяют класс различного типа языков проектирования и моделирования (VHDL, VERILOG, UML, GPSS).

Методическое обеспечение — описание технологии функционирования САПР, методов выбора и применения пользователями технологических приемов для получения конкретных результатов. Включает в себя теорию процессов, происходящих в проектируемых объектах, методы анализа, синтеза систем и их составных частей, различные методики проектирования.

Организационное обеспечение — совокупность документов, определяющих состав проектной организации, связь между подразделениями, организационную структуру объекта и системы автоматизации, деятельность в условиях функционирования системы, форму представления результатов проектирования.

Эргономическое обеспечение объединяет взаимосвязанные требования, направленные на согласование психологических, психофизиологических, антропометрических характеристик и возможностей человека с техническими характеристиками средств автоматизации и параметрами рабочей среды на рабочем месте.

Правовое обеспечение состоит из правовых норм, регламентирующих правоотношения при функционировании САПР, и юридический статус результатов её функционирования.

Будучи одной из сложных систем, САПР состоит из двух подсистем: проектирующей и обслуживающей. Примером первой подсистемы является геометрическое трехмерное моделирования механических объектов. С помощью обслуживающих подсистем осуществляется функционирование проектирующих подсистем. Характерными обслуживающими подсистемами считаются подсистемы управления процессом проектирования (DesPM – *Design Process Management)*, управления проектными данными (PDM – *Product Data Management*). Диалоговая подсистема; инструментальная подсистема; монитор, обеспечивающий взаимодействие всех подсистем и управление их выполнением — это обслуживающие подсистемы ПО. Диалоговая подсистема ПО даёт возможность интерактивного взаимодействия пользователя САПР с управляющей и проектирующими подсистемами ПО, а также подготовку и корректирование первоначальных данных, ознакомление с результатами проектирующих подсистем, функционирующих в пакетном режиме.

1.4. Классификация САПР

САПР классифицируют по следующим принципам: целевому назначению, по приложению, масштабам и характеру базовой подсистемы.

По целевому назначению выделяют:

- САПР-Ф или CAE (*Computer Aided Engineering*) системы, т.е. САПР функционального проектирования;
- САПР-К — конструкторские САПР общего машиностроения, чаще всего их называют CAD-системами;
- САПР-Т — технологические САПР общего машиностроения — АСТПП (*автоматизированные системы технологической подготовки производства*) или системы CAM (*Computer-Aided Manufacturing*).

По приложениям самыми важными и широко используемыми считаются такие группы САПР как:

- Машиностроительные САПР или MCAD (*Mechanical CAD*) системы — это САПР для применения в отраслях общего машиностроения;

- ECAD (*Electronic CAD*) или EDA (*Electronic Design Automation*) системы — САПР для радиоэлектроники;
- САПР в области архитектуры и строительства.

Помимо этого существует большое количество более специализированных САПР, или выделяемых в определенных группах, или являющихся самостоятельной ветвью в классификации. Это такие системы как: БИС САПР (*больших интегральных схем*); САПР летательных аппаратов и САПР электрических машин.

По масштабу определяют самостоятельные программно-методические комплексы (ПМК) САПР:

- Комплекс анализа прочности механических изделий в соответствии с методом конечных элементов (МКЭ);
- Комплекс анализа электронных схем;
- Системы ПМК;
- Системы с уникальными архитектурами программного (*software*) и технического (*hardware*) обеспечений.

Классификация по характеру базовой подсистемы:

- САПР, которые направлены на приложения, в которых главной процедурой проектирования является конструирование, то есть определение пространственных форм и взаимного расположения объектов. Это САПР на базе машинной графики и математического моделирования. К данной группе систем относится большая часть графических ядер САПР в сфере машиностроения;
- САПР, ориентированные на приложения, в которых при достаточно простых математических расчетах перерабатывается большое количество данных. Данные САПР главным образом встречаются в технико-экономических приложениях, например, В процессе проектирования бизнес-планов, объектов, подобных щитам управления в системах автоматики;
- Комплексные (интегрированные) САПР, которые включают в себя совокупность предыдущих видов подсистем. Типичными примерами комплексных САПР могут быть CAE/CAD/CAM-системы в машиностроении или САПР БИС;
- САПР на базе определенного прикладного пакета. По сути это свободно используемые программно-методические комплексы, такие как, комплекс имитационного моделирования производственных процессов, комплекс синтеза и анализа систем автоматического управления, комплекс расчета прочности по методу конечных элементов

и т. п. Как правило, данные САПР относятся к системам CAE. Например, программы логического проектирования на базе языка VHDL, математические пакеты типа MathCAD.

1.5. САПР интегральных микросхем

Развитие электроники неразрывно связано с развитием средств автоматизированного проектирования электронных устройств. Сегодня индустрия средств автоматизированного проектирования электроники (*Electronic Design Automation – EDA*) и в частности САПР интегральных микросхем (ИМС) – неотъемлемая часть электронной промышленности.

1.5.1. Предпосылки развития

Началом развития современной полупроводниковой электроники можно считать 50-е годы прошлого века. Первый точечный транзистор появился в 1947 году (рис. 1.2а), плоскостной – в 1950 году (рис. 1.2б), а первая интегральная микросхема – в 1959 году (рис. 1.2в). Первая советская германиевая интегральная схема 2НЕ-ИЛИ изготовлена в 1962 г. на Рижском заводе полупроводниковых приборов (рис. 1.2г). В центральном конструкторском бюро при Воронежском заводе полупроводниковых приборов (ВЗПП) (ныне АО «НИИЭТ») с июня 1965 г. началась научно-исследовательская работа по созданию кремниевой монолитной схемы (НИР «Титан»), которая была досрочно выполнена уже к концу года. Тема была успешно сдана Госкомиссии, и серия 104 микросхем диодно-транзисторной логики стала первым фиксированным достижением ВЗПП в области твердотельной микроэлектроники (рис. 1.2д). При появлении планарной технологии и первых планарных интегральных схем сформировалась технология проектирования, основные принципы которой остаются актуальными и в настоящее время.

Разработка первых ИС в 60-е годы велась вручную. Чертеж принципиальной электрической схемы выполнялся на бумаге вручную. На больших листах пленки из сложного полиэфира послойно реализовывались фрагменты проекта, которые потом объединялись на огромном листе бумаги (рис. 1.3а). Соответствие исходной принципиальной схемы и полученного проекта топологии проверялось визуально. Затем создавался комплект шаблонов для фотолитографии. Для этого каждый слой вручную вырезался

из специальной бумаги и фотографически уменьшался (рис. 1.3б). Даже первый микропроцессор Intel 4004 с 2300 транзисторами был спроектирован таким образом (рис.1.3в).

В 70-е годы стало очевидно, что при постоянном усложнении интегральных схем задача их промышленной разработки без создания средств компьютерной автоматизации будет нереализуема, и такие средства стали появляться. Инструменты автоматизации, которые сейчас объединены в рамках EDA, поначалу были представлены средствами CAE и CAD. Первые были призваны помочь разработчику принципиальных схем, вторые – инженеру-конструктору.

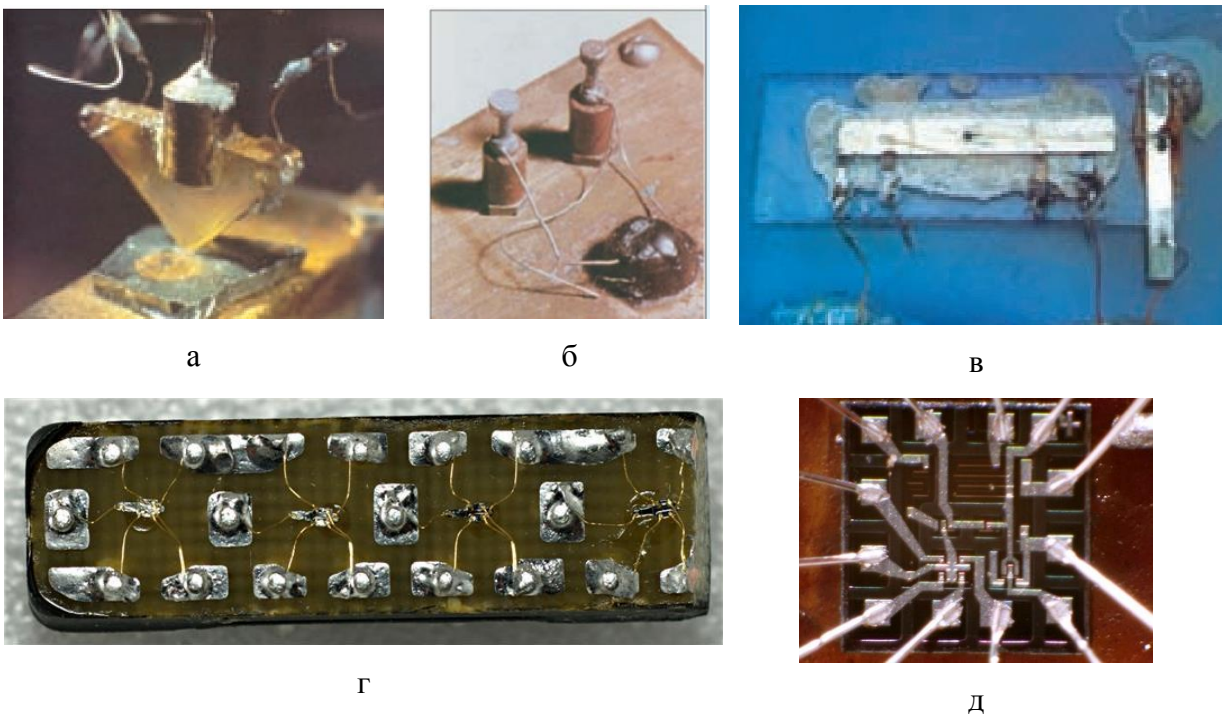
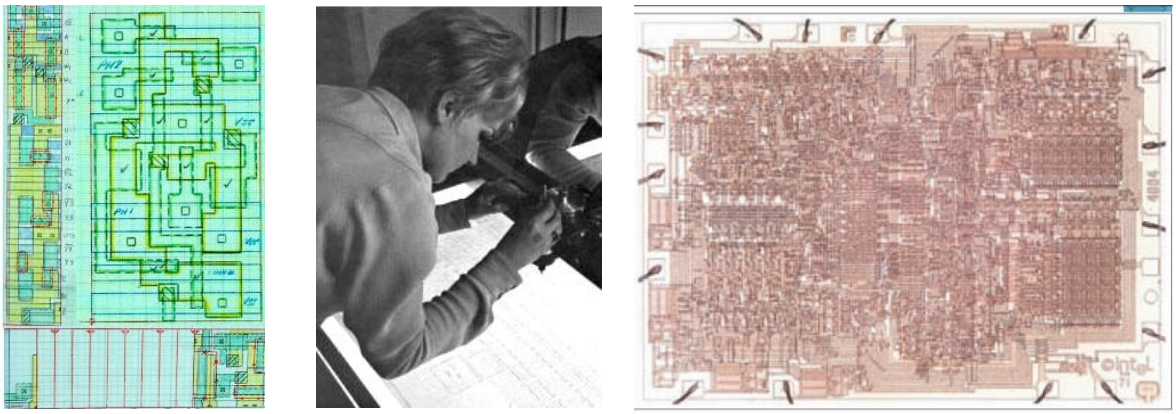


Рис. 1.2. Этапы развития современной полупроводниковой электроники: а – точечный транзистор (1947 г. AT&T Bell Laboratories); б – плоскостной транзистор (1950 г. AT&T Bell Laboratories); в – макет первой интегральной схемы (1959 г. Texas Instruments), первая советская интегральная схема (1962 г, Рижский завод полупроводниковых приборов); первая воронежская интегральная схема (1965 г., Воронежский завод полупроводниковых приборов)

Самой серьезной проблемой для разработчиков ранних ИС было отсутствие возможности создания физического прототипа разрабатываемого устройства. Ошибки, допущенные при проектировании схемы устройства, обнаруживались только после изготовления интегральной схемы. При обнаружении ошибки нужно было менять проект, заново создавать комплект фотошаблонов и повторять весь производственный цикл.



а б в

Рис. 1.3. Разработка первых ИС:
а – компоновочный чертеж на лавсановой пленке; б – перенос чертежа на рубилит;
в – микропроцессор Intel 4004 (1971 г., 2300 транзисторов)

Для решения этой проблемы в 70-е годы в университете Беркли (Berkeley), была разработана программа SPICE (*Simulation Program with Integrated Circuit Emphasis*), предназначенная для моделирования ИС на электрическом уровне. Таким образом, она позволяла проверять правильность работы схемы на уровне виртуальной компьютерной модели. Эта программа и по сей день используется для моделирования аналоговых схем. По мере распространения цифровых схем для проверки правильности функционирования стали разрабатывать и использовать средства логического моделирования. Одной из первых таких программ была система Hi-Lo.

В те же 70-е годы сформировались принципы организации системного программного обеспечения, применяющиеся и в современных EDA-средствах. Первые компоненты компьютерного проектирования (*CAE*) создавались на базе вычислительных машин коллективного пользования и только что появившихся мини-компьютеров. Но операционные системы, использовавшиеся в этих компьютерах, не удовлетворяли требованиям разработчиков. В лаборатории Bell Labs и в университете Berkeley были разработаны операционная система UNIX и язык программирования C. По сей день операционная система UNIX (различные версии) и язык C/C++ – самые популярные системные программные средства в EDA-индустрии.

Таким образом, появление средств CAD (компьютерных инструментов для конструкторского проектирования печатных плат и интегральных схем) в 70-е годы было обусловлено быстро растущей сложностью и объемом схем. Без таких инструментов уже

стало невозможно оптимально разместить и соединить между собой функциональные блоки сложных электронных устройств (рис. 1.4).

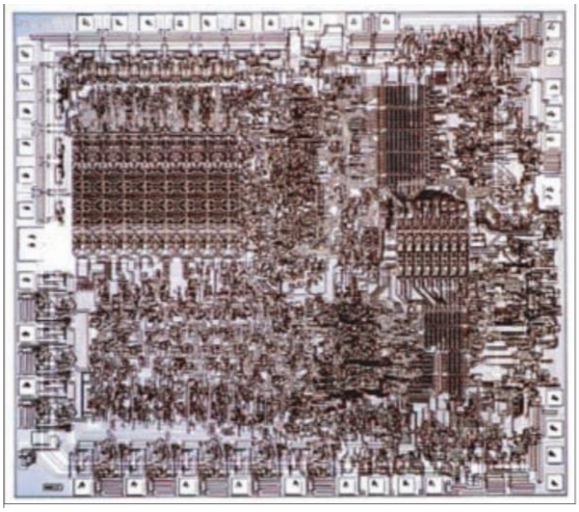


Рис. 1.4. Микропроцессор Intel 8080
(апрель 1974 г., 4758 транзисторов, технология 6 мкм)

Ведущие позиции в этой сфере принадлежали фирмам Computervision, Applicon и Calma (в 1988 году поглощена фирмой Valid, которая, в свою очередь, была полностью интегрирована в компанию Cadence в 1992 году). Разработанные фирмой Calma CAD-системы, такие как Startrec-Station, широко использовались вплоть до начала 80-х годов. Производительность этих систем была довольно ограниченной, а оборудование очень дорогостоящим, поэтому работы с применением этих средств CAD нередко велись круглосуточно в три смены.

1.5.2. Рождение индустрии САПР ИМС

В начале 80-х годов компании Daisy, Valid и Mentor Graphics разработали свои системы на базе рабочих станций, в рамках которых объединялись ввод принципиальной схемы, система моделирования и средства конструкторского проектирования. Таким образом, произошло объединение средств CAE и CAD. В 1985 году эти фирмы с большим успехом вышли на мировой рынок. Это и было рождением индустрии EDA.

Однако требования к средствам проектирования интегральных схем непрерывно возрастали. Это стимулировало развитие небольших начинающих компаний, выходящих

на рынок с новыми технологиями. В это время появились такие фирмы, как ECAD с системой Dracula, предназначенной для проверки соответствию правилам проектирования (*Design-Rule-Checking, DRC*) и контроля соответствия топологии исходной принципиальной схеме (*Layout-Versus-Schematic, LVS*), компания SDA с продуктом IC-Framework, компания Silicon Compiler Systems (*SCS*) с продуктом Silicon-Compiler, компания Tangent с системой размещения элементов и трассировки металлических соединений, а также компания Gateway Design Automation с языком Verilog и системой высокоуровневого логического моделирования на базе этого языка. Начались первые разработки технологических САПР, целью которых являлось моделирование как отдельных, так и совокупности технологических процессов. Таковым продуктом, появившимся за рубежом, является SUPREM(I, II, III), который разрабатывался в Стенфордском университете США. Данная программа содержала в себе описание основных технологических этапов: эпитаксия, диффузия, окисление, ионная имплантация, травление и т. д. Подобное программное обеспечение разрабатывалось в СССР, например, ПОМТИС, ТЕХИС1В, ДЕЛЬТА1. Первые технологические САПР позволяли получать лишь качественные результаты, так как в основе работы программ лежало одномерное моделирование.

Фирмы SDA и ECAD первыми осознали, что для завоевания рынка EDA необходимо обеспечить техническую поддержку объединения различных EDA-средств в рамках единой интегрированной среды проектирования. В 1988 году эти фирмы объединились в компанию Cadence. Среда SDA-Frameworks позволила собрать воедино различные перспективные технологии. Через год компания Cadence поглотила фирму Tangent и объединилась с фирмой Gateway, обеспечив тем самым лидерство как в области средств размещения/трассировки, так и в области средств логического моделирования. Вместе с компанией Gateway фирма Cadence приобрела права на язык Verilog и начала его продвижение на рынок, как в качестве специализированного языка программирования, так и в качестве средства моделирования цифровых систем. Впервые появилась возможность предложить разработчикам комплексное решение задачи проектирования интегральных схем.

В 1991 году компания Cadence смогла интегрировать в себя Valid, таким образом, удалось расширить сферу деятельности компании на область проектирования систем на печатных платах, обеспечив лидерство в EDA-бизнесе.

1.5.3. Особенности развития САПР цифровых и аналоговых ИМС

В конце 80-х годов интегральные микросхемы настолько усложнились, что создание описания принципиальной схемы, как с помощью схемотехнического редактора, так и в текстовом виде стало существенной проблемой. Были разработаны первые инструменты, которые позволяли из описания на уровне регистровых передач (*Register Transfer Level – RTL*, языки Verilog или VHDL) автоматически синтезировать описание принципиальной схемы на уровне логических элементов. В 1987 году компания Synopsys вышла на рынок с продуктом Design Compiler, за ней вскоре последовал AutoLogic от фирмы SCS-Mentor Graphics. Специалисты Synopsys вовремя оценили преимущества технологии автоматического синтеза для рынка СБИС на базе стандартных библиотек логических элементов (ASIC). Использование таких средств позволяло разработчикам проводить независимое проектирование ИС на верхнем уровне, осуществляя разработку описания на языках Verilog или VHDL и его верификацию средствами логического моделирования. После автоматического синтеза результаты проектирования в виде описания принципиальной схемы на уровне логических элементов передавались компании-производителю ИС. Компания-производитель выполняла физическое проектирование (размещение/трассировка, контроль правил проектирования и соответствия электрической схеме, подготовка данных для изготовления фотошаблонов), а также производство и тестирование. Таким образом, до середины 90-х годов на EDA-рынке в сфере физического проектирования и проектирования аналоговых и смешанных ИС лидировала компания Cadence, а в области логического синтеза господствовала Synopsys.

Последние десятилетия история развития EDA-индустрии отмечена в первую очередь успехами в сфере проектирования цифровых схем. В то же время средства аналогового, смешанного цифроаналогового и полностью заказного проектирования также постоянно улучшались как с точки зрения производительности, так и с точки зрения точности. Выход на рынок среды проектирования Virtuoso компании Cadence в начале 90-х годов позволил на порядок повысить производительность заказного проектирования. Среда Virtuoso постоянно пополнялась большим числом инновационных продуктов. Например, в 1993 году появилась система аналогового моделирования Spectre, в 1995 году были добавлены средства моделирования нелинейных радиочастотных (RF) схем, а в 2000 году – средства Verilog-AMS – единого инструмента для совместного цифро-

аналогового моделирования. Следующий шаг в сторону модернизации был сделан в 1997 году благодаря технологии IC-Craftsman.

1.5.4. Субмикронные технологии

Следом за полупроводниковыми технологиями развивались и технологические САПР. Уменьшение размеров элементов и разработка многослойных структур требовали построения двумерных моделей приборов. Для моделирования двумерных технологических процессов был разработан ряд технологических САПР: LADIS, BICEPS, SUPRA, SOAP, SAMPLE, SUPREM-III, ASPREM, ФАКТ2, ДЕЛЬТА2 и т.д. Основными признаками этих программ, как систем моделирования второго поколения являлись следующие возможности:

- моделирование процессов обработки многослойных структур со слоями Si, Si*, SiO2, Si3N4 (SUPREM-III, ASPREM);
- двумерное моделирование в непрямоугольной области (ФАКТ2, ДЕЛЬТА2);
- более точное моделирование процесса ионной имплантации;
- двумерное моделирование всех технологических операций;
- решение задачи движения границы SiO_2-Si.

В начале 90-х годов разработаны продукты следующего поколения, которые позволяли использовать модели диффузии с учетом неравновесных точечных дефектов, возникающих в процессе окисления, диффузии и ионной имплантации. Первопроходцем в создании продукта по трехмерному моделированию в технологической САПР стала фирма TOSHIBA. Была реализована трехмерная модель ионной имплантации методом Монте-Карло, позволяющая изучать траекторию внедренных ионов, распределение дефектов и эффект каналирования в приборах с субмикронными размерами. Коэффициент диффузии моделировался с учетом концентрационной зависимости и наличия других примесей. Была реализована модель боковой диффузии.

В середине 90-х годов появились первые интегральные схемы с нормами проектирования 0.5 и 0.35 мкм. Возникла необходимость учета различных тонких физических эффектов. Увеличилась суммарная длина проводников, вследствие чего выросли паразитные емкости и сопротивления, существенно выросли времена задержек. Определяющим фактором стало качественное проектирование топологии схемы. Компания Cadence

воспользовалась имеющимися наработками в сфере размещения/трассировки и представила разработчикам новый метод проектирования топологии (*back-end* – от технологии к решению), который позволял специалистам в области проектирования верхнего уровня (*front-end* – от решения к технологии) использовать технологии предварительного размещения.

Во второй половине 90-х годов с переходом на нормы проектирования 0.25 мкм возникла необходимость еще более тесной интеграции логического синтеза и проектирования топологии ИС. Компания Cadence использовала свою систему синтеза Ambit для разработки программы PKS Physical Synthesis, объединяющей синтез и размещение. Вскоре подобный продукт под названием Physical Compiler был выпущен и фирмой Synopsys.

С переходом на субмикронные технологии стали всё чаще появляются отдельные независимые дизайн-центры (*fabless company*), которые приобретали EDA-средства для обеспечения процесса проектирования ИС (*Customer Own Tooling*, так называемая COT-модель). Дизайн-центры осуществляют полный цикл разработки ИМС с использованием собственной методологии и IP-библиотек, а заказы на производство размещают на таких фабриках, как TSMC, UMC и Chartered. С возникновением этой бизнес-модели для EDA-индустрии открылась ещё одна рыночная ниша, в которой можно было предлагать не только продукты, но и технологии проектирования и сервисные услуги.

1.5.5. Нанометровые технологии

На рубеже тысячелетий началось производство с применением технологических норм 130, 90 и 65 нм. При использовании таких технологий решающими факторами успеха становится учет паразитных явлений и оценка искажения сигнала. Это означает, что должна учитываться реальная конфигурация соединений. Средств физического синтеза уже недостаточно для получения требуемого результата. Возникла необходимость разработки новых методов. В 2002 году компания Cadence поглотила фирму Silicon Perspective, реализовавшую в продукте First Encounter концепцию виртуального прототипа. Система First Encounter фактически стала стандартом для проектирования с использованием нанометровых технологий.

В условиях жесткой конкуренции и гонки технологий компании-производители постоянно совершенствуют свои технологические процессы. На данный момент определены прогнозы по переходу на 5 и 6 нм технология к 2021 г (табл. 1.1).

В настоящее время EDA-индустрия развивается с учётом требований нанометровых технологий и методологии проектирования "система на кристалле" (*systems-on-chip, SoC*). Системное проектирование, проектирование схем малой мощности, проектирование с учетом требований производства, квантовых эффектов, интеграция систем – вот лишь некоторые из тех областей, которым разработчики в ближайшие годы должны будут уделить пристальное внимание. В любом случае можно утверждать, что продолжение будет интригующим.

Таблица 1.1

План развития технологий (топологические нормы в нм) крупнейших производителей полупроводниковых ИМС по полугодиям (DUV – глубокий ультрафиолет, длина волны 193 нм, EUV – экстремальный ультрафиолет длина волны ~ 10 нм)

<table>
<tr><th rowspan="2">Годы
Фирма</th><th rowspan="2">2016</th><th colspan="2">2017</th><th colspan="2">2018</th><th colspan="2">2019</th><th colspan="2">2020</th><th rowspan="2">2021</th></tr>
<tr><th>1 пг</th><th>2 пг</th><th>1 пг</th><th>2 пг</th><th>1 пг</th><th>2 пг</th><th>1 пг</th><th>2 пг</th></tr>
<tr><td>Global Foundries</td><td colspan="3">14</td><td colspan="2">7 (DUV)</td><td colspan="4">7 (EUV)</td></tr>
<tr><td>Intel</td><td colspan="3">14
14+</td><td colspan="2">14++
10</td><td colspan="4">10+
10++</td></tr>
<tr><td>Samsung</td><td>14</td><td colspan="3">10</td><td>10</td><td colspan="2">8</td><td colspan="2">7</td><td>6</td></tr>
<tr><td>SMIC</td><td>28</td><td colspan="9">14</td></tr>
<tr><td>TSMC</td><td>16+
16</td><td colspan="3">10
16</td><td>7
12</td><td>12</td><td colspan="2">7</td><td>5</td></tr>
<tr><td>UMC</td><td colspan="2">28</td><td colspan="2">14</td><td colspan="6">нет данных</td></tr>
</table>

Стремительный прорыв в области полупроводниковых технологий был бы не возможен без развития средств приборно-технологического проектирования TCAD. На данный момент существует ряд продуктов, например, Silvaco TCAD, TCAD–CEDA, NovaTCAD, Lumerical DEVICE, MicroTec, Sentaurus (ISE) TCAD, позволяющих провести моделирование технологических этапов и исследовать его электрофизические параметры. Данные САПР позволяют моделировать технологические операции в 1D, 2D и 3D пространстве, для разработки и оптимизации технологических этапов кремниевых полупроводниковых приборов и предоставляет прогностическую основу для моделирования

широкого спектра технологий, от наноразмерных КМОП до крупномасштабных высоковольтных силовых устройств.

1.6. Будущее САПР

На сегодняшний день «облачные» сервисы существенно продвинулись в сегменте легких приложений преимущественно в потребительском секторе. Поэтому одна из ключевых тем развития САПР - облачные вычисления, т.е. работа с данными, размещенными на удаленных серверах, с различных устройств через интернет. Возможны два варианта интеграции. В первом случае в облако переносится вся инфраструктура инженерных служб, и соответственно исчезает необходимость в инженерном ПО, установленном на рабочем месте. Во втором случае у конструктора остается графическая рабочая станция с установленной САПР, через которую он получает доступ к различным облачным сервисам, благодаря которым можно решать задачи, требующие весьма существенных ресурсов.

Следующая важная тенденция – альтернативные операционные системы. Еще несколько назад серьезной альтернативы Microsoft Windows или Linux не было. Однако сейчас можно говорить о существенном потенциале операционной системы Google Chrome OS, которая одновременно обеспечена облачным сервисом ОС Google. Если в облака перенести большинство громоздких и сложных вычислений, снижаются требования к аппаратному обеспечению и появляется возможность работать на любых устройствах, например, на планшетах. В итоге разработчикам САПР-решений придется либо разрабатывать платформонезависимые решения (облачный вариант), либо делать их мультиплатформенными.

Следующая тема определяется неудовлетворенностью рынка классической архитектурой процессоров Intel и темпами ее развития. Отмечается тренд на развитие архитектуры ARM. Ее сейчас поддерживает несколько производителей, одним из самых активных является компания Nvidia. Пока данная архитектура активно применяется в мобильных устройствах, но в ближайшее время ожидается массовый переход на стационарные ПК. Подход заключается в переносе существенной части вычислений с центрального процессора на графическое ядро и включает в себя теорию параллельных вычислений.

В секторе САПР сегодня многие сотрудники являются мобильными, т.е. активно перемещаются по стране и миру, могут работать из разных точек, в том числе из дома. Все это требует удобного мобильного устройства. Уже появились привлекательные для разработчиков мобильные платформы IOS Apple и Android Google, а также существенное количество САПР-приложений под них. При этом возникает интересная ситуация: мультитач-экраны практически стали стандартом в мобильных устройствах, так как этот интерфейс подходит для потребления информации. Так же ли он хорош для ее создания при работе в САПР – покажет время.

Следует учитывать, что замена одной системы на другую в рамках работы над одним проектом довольно сложная задача. Поэтому рынок САПР консервативен и явно не входит в число лидеров технологической гонки. Развитие есть, но не такое быстрое как в пользовательском сегменте. Впрочем, в ближайшее десятилетие на предприятия придут инженеры, выросшие уже в эпоху интернета, новых технологий и мобильных устройств, и так или иначе они станут активно привносить на рынок элементы своей культуры.

1.7. Вопросы для самопроверки

1. Что такое САПР, CAD-, CAM-, CAE-, EDA-system? Чем они отличаются?
2. Цели и задачи САПР? Какой у Вас есть опыт работы в САПР?
3. Области применения САПР. Какие задачи решают с помощью САПР в них?
4. Перечислите виды обеспечения САПР.
5. Особенности проектирования ИМС до и после появления САПР ИМС.
6. Что определяло направления развития САПР ИМС?
7. Какие задачи решают в САПР ИМС в микронных, субмикронных и нанометровых технологиях?
8. В каких направлениях развиваются САПР?

2. ВВЕДЕНИЕ В МЕТОДОЛОГИЮ ПРОЕКТИРОВАНИЯ

2.1. Введение

С определения требований к ИМС как к конечному продукту для использования начинается ее проектирование. Далее определяются принципиальная возможность ее изготовления, архитектура, используемая технология, размер кристалла, тип корпуса, возможность использования имеющихся наработок, временные затраты, итоговая стоимость и в конечном счете экономическая целесообразность разработки – это небольшой перечень задач, стоящих перед разработчиками. Современное проектирование ведется по спиралевидному типу, что означает, что разработка программного обеспечения, поведенческого описания, логического и физического синтеза ведутся параллельно, постоянно дополняя и уточняя друг друга. При такой системе важным является соблюдение методологии проектирования и четкое соблюдение маршрута проектирования.

Из данной главы вы узнаете:

- ✓ *С чего начинается проектирование ИМС*
- ✓ *Что такое ниспадающая и спиралевидная методология проектирования*
- ✓ *Как разработка ПО, RTL-кода, логическая и физическая верификация могут проводиться одновременно*
- ✓ *Этапы проектирования*
- ✓ *Виды и влияние верификации на качество ИМС*
- ✓ *Что включает в себя заключительная верификация ИМС*
- ✓ *Можно ли спасти проект, если ошибка в ИМС обнаружена после запуска на фабрику*
- ✓ *Как компания Freescale Semiconductor завоевывает рынок автомобильной промышленности*

2.2. Исходные данные для проектирования ИМС

Первоначальными исходными данными для разработки ИМС являются:

- ➢ перечень технических параметров (быстродействие, режимы эксплуатации, разрядности входных данных);
- ➢ формализованная задача;

- структурное описание;
- алгоритмы функционирования;
- необходимые интерфейсы;
- спецификация программного обеспечения.

На основании этих данных разработчик формирует структурную схему проекта и определяет, какая часть алгоритма будет выполняться аппаратно, а какая программно, какие интерфейсы необходимо разработать, какие потребуются сценарии моделирования.

Затем необходимо провести предварительную оценку быстродействия, потребляемой мощности и логической емкости проекта для декомпозиции проекта и выбора необходимых модулей с целью обеспечения требуемых в техническом задании характеристик. После проведения такого анализа формируются спецификации проекта для разработчиков схем электрических, программистов прикладного и системного ПО, специалистов по моделированию и верификации.

В спецификацию проекта **для разработчика схемы электрической принципиальной** включают следующие характеристики:

- логический объем, быстродействие, типа корпуса;
- группы используемых выводов, с указанием их функционального назначения;
- топологические характеристики, технологические требования;
- необходимость организации внешних интерфейсов и применения IP блоков;
- требования к трактам обработки данных (например, требования к АЦП и ЦАП, необходимости внешнего ОЗУ, процессора);
- требования к тактовой частоте (стабильность, форма сигнала);
- требования к системе питания (максимальный потребляемый ток по всем питаниям и землям, порядок включения).

Для конструкторов спецификация проекта включает следующие требования:

- тип корпуса;
- группы используемых выводов с указанием топологических к трассировке сигналов;
- оценка потребляемой мощности для определения температурного режима и разработки радиатора.

Для программистов прикладного и системного ПО указывают:

- алгоритм обработки данных в проекте, с указанием какая часть будет реализована программно;
- тип используемого процессора и способ его реализации («аппаратно», «проектно» или является «внешним»);
- алгоритм (протокол) взаимодействия между модулями и процессором, включающий в себя карту памяти, обслуживание прерываний, механизмы обмена данными;
- необходимость конфигурировать специализированные интерфейсные микросхемы;
- состав и функции прикладного ПО;
- тип операционной системы и необходимые ее функции, если планируется ее применение.

Специалисту по тестированию в спецификации указывают:

- алгоритм обработки данных, с указанием, что реализуется с помощью ПО, а что аппаратно;
- перечень интерфейсов и версий спецификаций, поддерживаемых проектом;
- алгоритмы и протоколы взаимодействия с ПО.

На основе полученных исходных данных параллельно разрабатывается проект, прикладное и системное ПО, схема электрическая принципиальная, топология, план верификации и набор тестов.

2.3. Маршрут проектирования ИМС

В настоящее время на смену традиционному ниспадающему («*waterfall*») типу проектирования пришла спиралевидная методология проектирования. В ниспадающей модели, перед тем как перейти к следующему этапу, должны быть завершены все проектные задачи на текущем этапе. В более реалистичной спиралевидной модели проектирование выполняется одновременно по 4-м направлениям:

1. разработка программного обеспечения;
2. разработка RTL-кода;
3. логический синтез;

4. физический синтез;

При этом в процессе работы группы разработчиков обмениваются результатами проектирования. Например, при разработке архитектуры ИМС может возникнуть необходимость в полной или частичной реализации IP-блока (*intellectual product*, готового полностью верифицированного и проверенного «в кремнии» блока), чтобы оценить его размеры, потребляемую мощность или производительность, и по результатам такой оценки архитектура может быть частично или полностью модифицирована. Подобная концепция быстрого прототипирования части проекта или всей ИМС на архитектурном уровне является неотъемлемой частью современных маршрутов проектирования.

Определим каждый этап процесса проектирования.

Анализ требований микросхемы - это определение и описание сложной микросхемы, как конечного продукта, с точки зрения ее необходимости для радиоэлектронной аппаратуры (РЭА), где она будет эксплуатироваться. Исходной информацией является набор характеристик будущей микросхемы, полученных в результате маркетинговых исследований: стоимость, размеры, потребление энергии, производительность, тип корпуса и т.п. Данный этап должен дать однозначный ответ на вопрос: является ли проект осуществимым? Какие усилия необходимо приложить, чтобы осуществить проект? Какая часть проекта может быть сведена к повторному использованию? Может ли данный проект быть построен на основе аналогичных проектов продуктов предыдущего поколения?

Разработка и анализ алгоритма. На данном шаге создаются, адаптируются и исследуются ключевые алгоритмы работы микросхемы, как в части управления, так и в части обработки данных. Алгоритмы управления часто выражаются в форме автомата с конечным числом состояний. Алгоритмы обработки данных, как правило, записываются на языке специализированных программных средств, в частности, на языке C/C++. Для анализа такого рода структур данных могут быть использованы следующие автоматизированные средства: Mathworks Matlab/ Simulink, Cadence SPW, Synopsys CoCentric System Studio и Ptolemy/ Ptolemy II.

Основные вопросы, ответы на которые должны быть получены на данном этапе: какие типы обрабатываемых данных и какие управляющие функции необходимы для нормальной работы данной микросхемы? С какой точностью должны обрабатываться данные? Для систем связи, например, необходимо определить какие методы и алгоритмы кодирования/декодирования будут использованы. Можно ли адаптировать суще-

ствующие алгоритмы или необходимо создавать новые? Способен ли выбранный алгоритм обеспечить требуемую пропускную способность? Насколько достоверна, используемая идеализированная модель канала? Насколько сложен выбранный алгоритм в реализации на архитектурном уровне?

Архитектура. На данном этапе определяется базовая архитектура будущей микросхемы. В первую очередь нужно определиться с программируемыми процессорными ядрами. Здесь разработчик должен ответить на следующие вопросы: Будет ли использоваться процессор? Сколько ядер необходимо? Будет ли интегрироваться на кристалл только процессор или готовая процессорная подсистема? Процессорные ядра часто комплектуются целой подсистемой жизнеобеспечения, состоящей из дополнительного набора IP-блоков (интерфейсы, контроллеры памяти, устройства диагностики и т.п.). Каковы идеи по разбиению на программную и аппаратную части? Какую структуру памяти предпочтительней использовать? Какие требования должны предъявляться к блокам встроенной памяти (размеры, быстродействие, доступ).

Проектирование на системном уровне. На системном уровне анализируют характеристики и производительность системы, проверяют архитектуру, определяют способы разрешения конфликтов, оптимизируют алгоритмы и протоколы. При необходимости вносят функциональные изменения в отдельные блоки, перераспределяют программные и аппаратные средства и повторно моделируют работу системы. В итоге должна быть решена задача программно-аппаратной декомпозиции. В среде Cadence для этих целей используются такие инструменты как SPW2000 и i-Architect.

В систему SPW2000 (*signal processing worksystem* – рабочая среда обработки сигналов) входят средства моделирования системного уровня (включая поддержку моделей на языках C/C++, SystemC, Verilog, VHDL, VerilogAMS, а также созданных в среде Matlab), системные библиотеки элементов, виртуальные генераторы и анализаторы сигналов, инструментарий разработки фильтров, средства эмуляции системы.

Сама модель строится в специализированном графическом редакторе BDE (*Block Diagram Editor*) в виде иерархической блок-диаграммы – от общего описания системы до дефрагментации ее на отдельные функциональные блоки. Каждый блок описывается поведенческой моделью и набором требуемых характеристик – спецификацией блока. Стандартные библиотеки содержат множество готовых модулей, например: системы сотовой связи WCDMA, cdma2000, GSM/GPRS/HSCSD/EDGE; беспроводные сети стандартов IEEE802.11a/b, HiperLAN/2, Bluetooth; телевизионные системы стандартов DVB-

T, ISDB-T, NTSC; модемы физических линий; библиотеки высокочастотных систем и радаров, видео кодеки, датчики, системы компрессии и т.д. Библиотеку SPW2000 можно дополнять собственными, созданными на языках C/C++/SystemC, и модели, созданные в среде Matlab.

Принципиально, что при моделировании система может включать блоки, описанные на разных уровнях представления, как на системном, так и на уровне регистровых передач (RTL-уровень), на языках VHDL/Verilog. Таким образом, в системной модели можно использовать непосредственно описания IP-блоков. После выбора IP-блоков, требуемых для проекта, необходимо предпринять действия по их приобретению. Одна из ключевых трудностей – это проведение строгой и исчерпывающей проверки приобретаемых IP-блоков на предмет их достоверности и полноценности, а также выявление возможных ошибок до этапа интеграции.

Данный подход существенно сокращает число циклов проектирования и повышает вероятность успешной реализации проекта с первого запуска на фабрику. В результате проектирования на системном уровне формируются детальное описание архитектуры системы, спецификации отдельных программных и аппаратных блоков и их функциональное описание на языках C/C++ и SystemC.

Верификация на уровне транзакций. Поведенческая модель, построенная на этапе системного проектирования, может служить основой для создания более детализированной модели, использующей абстракции уровня транзакций. Модель уровня транзакций позволяет построить виртуальный прототип системы для проведения функциональной верификации программной и аппаратной частей проекта на уровне архитектуры. Виртуальный прототип позволяет верифицировать как полную микроархитектуру проекта, так и его отдельные части, описанные на языках (Verilog, VHDL).

Определение архитектуры встраиваемого ПО. Специфика области электроники, в которой будет эксплуатироваться разрабатываемая микросхема, оказывает значительное влияние не только на выбор архитектуры или процессорного ядра, но и ПО. В частности, выбор базовой операционной системы реального времени (*RTOS - Real-Time Operating System*) напрямую зависит от разрядности процессора, который, в свою очередь, определяется конкретной областью применений будущей микросхемы. Периферийные устройства должны быть обеспечены драйверами – специализированным ПО, которое обеспечивает взаимодействие прикладного ПО, RTOS и аппаратной периферии. Различные готовые пакеты специализированных программ – важная часть архитектуры встраиваемо-

го ПО. Таким образом, при создании ИМС необходимо четко и полно определить всю архитектуру программного обеспечения. Какую выбрать операционную систему? Как организовать взаимодействие ОС, прикладного ПО и периферии? Какие программные блоки можно получить со стороны и использовать повторно, а какие придется разрабатывать полностью?

Конфигурация и планировка микроархитектуры ИМС. На этом этапе мы начинаем рассматривать ИМС более детально с точки зрения логического и физического проектирования. Перед этапом детализации проекта до физического уровня должны быть выполнены следующие условия:

- достигнуто соглашение между членами команды по поводу физической планировки кристалла;
- все IP-блоки полностью и окончательно сконфигурированы;
- определена и сконфигурирована микроархитектура (тесты, питание, тактовые сигналы, шины, временные параметры);
- все аппаратные блоки должны быть реализованы (как минимум в виде RTL-кода).

Кроме этого, необходимо сгенерировать тестовое окружение, которое будет в дальнейшем использоваться для верификации ИМС программным путем.

Архитектура DFT и ее внедрение. Дополнительные схемы для тестирования и самотестирования готовых ИМС (*DFT - Design-for-Test*) должны быть обязательно встроены в проект. Положение затрудняется использованием готовых IP-блоков. Различные IP-блоки от разных поставщиков и производителей, как правило, имеют разные средства самотестирования, поэтому при разработке ИМС придется комбинировать все эти средства (такие как, BIST – *Build-in Self-Test*, или SCAN). Можно адаптировать различные тестовые средства к стандартному тестовому протоколу (например, JTAG) и разработать объединенный тестовый план.

Реализация и внедрение AMS блоков. Многие ИМС включают в себя аналоговые и смешанные (цифро-аналоговые) блоки (AMS - *Analog/Mixed Signal*), чтобы обеспечить связь с внешним миром. Существует пакет документов с рекомендациями по созданию и интеграции IP-блоков в ИМС. Данный пакет ориентирован на ИМС типа «Big D/Little A» (т.е. большая часть - цифровая, меньшая - аналоговая).

Интеграция аппаратных IP-блоков. Чтобы избежать трудностей следует проводить тщательную проверку и отбор поступающих IP-блоков.

Ассемблирование ПО и его интеграция. IP-блоки с программным обеспечением точно также, как и чисто аппаратные IP-блоки, должны быть адаптированы к конкретной ИМС, интегрированы и верифицированы в составе ИМС. Очень важно верифицировать работу программного IP-блока именно в контексте задач, решаемых всей ИМС.

Программно-аппаратная верификация. Это одна из самых трудоемких и ресурсоёмких задач, от решения которой будет зависеть качество всей ИМС. Чтобы верификация была эффективной необходимо использовать результаты по разработке тестового окружения, полученные на предыдущих этапах. Только хорошая организация процесса верификации позволит получить точный ответ на вопрос: «Как нам узнать, что верификация выполнена?» и перейти к дальнейшей работе.

Фаза заключительной сборки и верификации ИМС включает в себя: размещение элементов, трассировку всего кристалла, физическую верификацию, состоящую из контроля технологических норм DRC (*Design Rule Checking*), проверки соответствия топологии и схемы (LVS – *Layout vs. Schematic*) и экстракции паразитных элементов RCX (*Resistor Capacitor extraction*).

Весь комплекс задач по физическому прототипированию может решать платформа Cadence First Encounter Ultra. Платформа включает: средства физической оптимизации, автоматическое разбиение кристалла, присвоение выводов; иерархический синтез сигнала синхронизации; планирование сетки питания; интерфейсы с ведущими средствами разработки топологии и удобный пользовательский интерфейс.

Схемы, изготавливаемые по технологиям глубокого субмикрона (DSM - *Deep Submicron*) 0.18 мкм и меньше, должны проходить анализ на предмет отсутствия в них характерных для DSM эффектов: падение напряжения на шинах земли/питания (*IR-drop*), целостность сигнальных цепей (*signal integrity*); целостность цепей земли/питания (*power network integrity*). Для решения комплекса этих задач могут быть использованы средства Cadence, ориентированные на нанотехнологии: Fire&Ice – 3D-экстракция паразитных параметров из топологии кристалла; Celtic – быстрая и эффективная система анализа перекрестных помех на проводниках и искажений сигналов; VoltageStorm – инструмент учета влияния разброса напряжения питания на временные характеристики системы.

Производство, тестирование, корпусирование и лабораторные испытания. После того как ИМС передана на производство необходимо более детально проработать и проверить ПО в контексте аппаратной части ИМС. На практике иногда применяют подход,

когда отдельные ошибки в аппаратной части могут быть локализованы и устранены с помощью ПО. Когда протестированный и закорпусированный на заводе опытный образец микросхемы передается для окончательной верификации в лабораторию, идеальный сценарий для проверки – это загрузка ПО и тестирование всей системы целиком в течение нескольких часов. Следует отметить, что большинство продвинутых команд разработчиков с хорошо организованной методологией и маршрутом проектирования регулярно добиваются успеха в разработке ИМС без дополнительных незапланированных затрат.

2.4. Пример: разработка датчика давления в шинах

Автоиндустрия – очень крупный потребитель электронной продукции. Современные автомобили буквально напичканы электроникой. Поэтому появление требований или инициатив со стороны государств или автоассоциаций, направленных на повышение безопасности и оснащенности автомобилей, воспринимается производителями электронной техники с большим интересом. Рассмотрим конкретный пример – датчик контроля давления в шинах (TPMS – *Tire Pressure Monitoring System*). TPMS системы призваны сообщать водителю о снижении давления в одной или нескольких шинах для сохранения высокой управляемости автомобиля, снижения расхода топлива, предотвращения перегрева колеса, уменьшения износа протектора. На территории США и многих европейских стран действует закон об обязательном оснащении автомобилей данными датчиками и составлен список требований, которым такие системы должны соответствовать:

- точность порядка 2% полной шкалы в температурном диапазоне от 0 до 70 °C;
- высокая надежность в условиях повышенных температур, ударов, вибраций;
- минимальный срок службы порядка 6 лет/100 000 км;
- допускается мониторинг запасной шины;
- тестирование выполняется на скоростях >25 км/ч;
- предупреждающий сигнал о давлении подается в течение 3 мин;
- предупреждение о сбое в течение 10 мин;
- допускается прямой и непрямой контроль давления;

- соответствие TPMS требованиям Федеральной комиссии по связи FCC (*Federal Communications Commission*) и Европейского Института Стандартов для коммуникаций ETSI (*European Telecommunications Standards Institute*). Рабочие радиочастоты выбираются в нелицензированных частотных диапазонах ISM 315 – 434 МГц.

Таким образом, в самом общем виде задача уже формализована. Разработчикам необходимо определить детали и выполнить декомпозицию проекта. В частности, определиться с методом контроля: непрямой метод заключается в измерении угловой скорости (у спущенного колеса она будет выше); прямой метод заключается в непосредственном измерении давления в шинах и передачи информации на бортовой компьютер. Основное преимущество косвенного метода – это низкая цена и простота реализации. Недостатками являются: невозможность определять давление шин до начала движения; отсутствие возможности диагностировать состояние запасной шины; нельзя определять давление шин в узлах с двойными колесами; существуют ограничения в скорости, ускорении и траектории движения; способны детектировать «недокачку» только более 30%.

Если у разработчика имеется большой опыт в создании микроконтроллеров, радиопередатчиков, микроэлектромеханических систем (МЭМС), в частности интегральных МЭМС датчиков давления и ускорения, то есть все шансы сделать конкурентно способное изделие с прямым методом контроля давления. Стандартная архитектура системы TPMS при этом включает (рис. 2.1):

1. Четыре колесных модуля измерения давления, содержащие:

1.1. датчик измерения давления (КМОП/МЭМС);

1.2. датчик температуры;

1.3. блок формирования сигнала и идентификации шины на основе микроконтроллера и памяти;

1.4. трансмиттер, передающий радиосигнал RF-диапазона на приборную панель;

1.5. RF-антенну;

1.6. кристаллический (кварцевый) резонатор;

1.7. батарею;

2. Приемник в приборной панели;

3. Блок обработки сигнала в приборной панели.

Данную архитектуру применила компания Freescale Semiconductor. На основе данной архитектуры определены требования для трех ключевых модулей (табл. 2.1). Да-

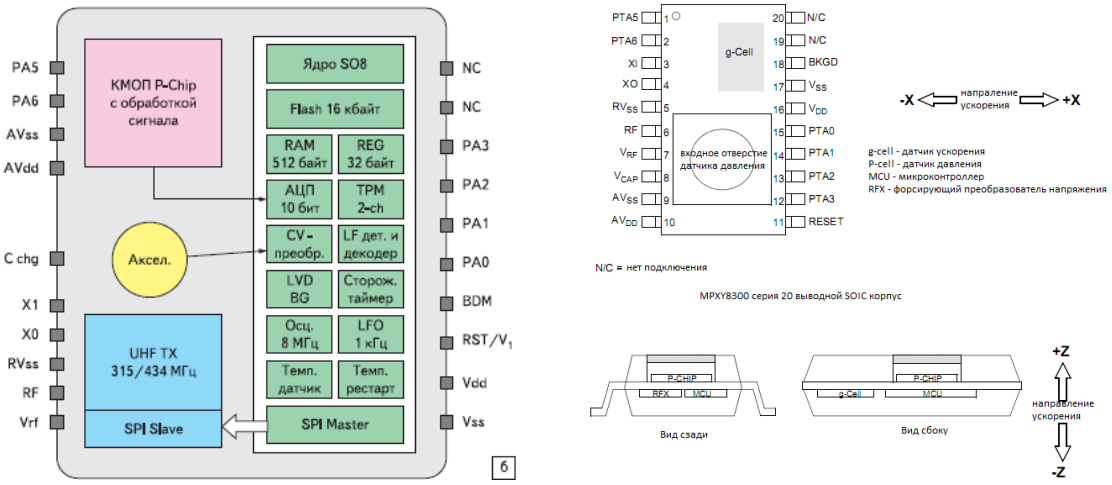
лее определяются требования и состав блоков, возможность использования уже имеющихся наработок для каждого из модулей. Компания имела налаженное производство микроконтроллеров, датчиков давления, акселерометров и радиопередатчиков. Поэтому для колесного модуля оптимальным решением являлось объединение в одном корпусе имеющихся кристаллов, взаимное расположение которых представлено на рисунке 2.2. Затем определена распиновка и тип корпуса, протокол передачи данных. Уделено внимание значительному снижению энергопотребления, так как минимальный срок автономной службы модуля должен составлять не менее шести лет. С этой целью система оснащена сторожевым таймером, который периодически посылает сигнал на включение микроконтроллера для обработки данных с датчиков, и далее микроконтроллер выключается.

Рис. 2.1. Архитектура TPMS системы с методом прямого контроля давления в шинах и пример колесного модуля MPXY8300 от Freescale Semiconductor

Т а б л и ц а 2.1

Характеристики модулей TPMS системы

Колесный модуль	Приемник	Низкочастотный передатчик
– 8-битный микроконтроллер; – оперативная память 512 байт; – флэш память 8 кбайт; – таймер слежения; – внутренний генератор; – датчик давления; – датчик температуры; – Z-осевой и X-осевой акселерометр; – радиочастотный передатчик 315/434 МГц; – источник питания 2.1...3.3 В	– 315-434 МГц; – область приема 20 м; – скорость передачи данных 9.6 кбит/с; – соединение по формату USB; – возможность перепрошивки микроконтроллера	– 8-битный процессор; – область приема 0.5 м; – частота 125 кГц. – НЧ-блок для уменьшения потери мощности



а б

Рис. 2.2. Колесный модуль MPXY8300: а – структурная схема; б – взаимное расположение кристаллов в корпусе

2.5. Вопросы для самопроверки

1. Перечислите исходные данные для разработчика ИМС.
2. Что включает в себя спецификация разработчика схемы электрической принципиальной?
3. Какая информация необходима конструктору и специалисту по тестированию?
4. Перечислите типы проектирования. Чем они отличаются?
5. Перечислите основные этапы проектирования.
6. Какие типы верификации проходит ИМС при спиралевидном типе проектирования?
7. Зачем проводить верификацию ИМС после запуска в производство?
8. Кто определяет формализованные задачи разработчикам ИМС?

3. ТИПОВОЙ КОМПЛЕКТ РАЗРАБОТКИ

3.1. Введение

Особенностью проектирования ИМС является привязка к конкретной технологии. Если фабрика-изготовитель заинтересована в том, чтобы как можно больше разработчиков пользовалось ее производственными мощностями, то она создает и оснащает разработчиков комплектом разработки. Комплект разработки включается в себя описание технологии с ее возможностями и ограничениями, правила проектирования, списки используемых слоев и доступных устройств с их моделями для обеспечения возможности верификации схем. Библиотеки элементов содержат различные представления, обеспечивая возможность использования одной технологии для решения разных задач в многочисленных программных пакетах. Серьезные фабрики-изготовители стремятся к совершенствованию своих технологических процессов, повышению точности математических моделей, расширению элементной базы, что ведет к необходимости периодического обновления комплектов разработки. В связи с этим важна синхронизация данных, имеющихся у разработчиков и изготовителей, так как ИМС, выполненная в полном соответствии с устаревшими правилами проектирования, может иметь серьезные ошибки с точки зрения новых правил.

Из этой главы вы узнаете:

- ✓ *Что такое PDK*
- ✓ *Функции PDK*
- ✓ *За счет чего обеспечивается автоматический синтез сложно функциональных ИМС*
- ✓ *Как один элемент может иметь девять представлений*
- ✓ *Как изменить параметры элементов*
- ✓ *Почему в PDK может содержаться пять слоев первого уровня металла*

3.2. Функции PDK

Типовой комплект разработки (*PDK, Generic Process Design Kit*) является одним из четырех базисов, на котором строится среда проектирования или платформа разработки ИМС. Помимо PDK для разработки необходимы маршрут проектирования, ин-

струменты проектирования, библиотеки.

PDK содержит в себе формализованное для разработки описание технологического процесса, интегрированного в среду проектирования. Это описание представлено в виде технологических библиотек, шаблонов, описаний слоев, устройств и их моделей, правил проектирования и верификации.

PDK выполняет следующие основные функции:

1. Обеспечивает базис для разработки и тестирования ИМС.
2. Предоставляет разработчику маршрут проектирования и примеры проектов.
3. Обеспечивает технической поддержкой при разработке.
4. Улучшает тестопригодность инженерных решений в проекте и предсказуемость процесса разработки.
5. Предоставляет инженерам доступ к консультативной базе по вопросам проектирования, использования САПР, PDK и технологий.
6. Обеспечивает лицензионную чистоту.
7. Обеспечивает легитимность процесса разработки.

В составе САПР Cadence имеется типовой комплект разработки GPDK090 (БиКМОП технология 90 нм), описывающий «поведенческий», не реализованный на какой-либо фабрике технологический процесс. Тем не менее, PDK полностью поддерживает методологию проектирования CIC «сверху-вниз» и содержит все необходимые устройства, модели, слои, правила проектирования и верификации.

3.3. Представления элементов

Существующие в настоящее время средства САПР ИМС позволяют производить автоматический синтез сложнофункциональных микросхем на основе заранее спроектированных библиотек элементов. Такие библиотеки содержат схемотехнические и топологические представления элементов:

symbol – представление условно-графическим символом;

spectre – список соединения для моделирования в Spectre и UltraSim;

schematic – список соединений для всех симуляторов;

hspiceD – список соединений для симулятора Hspice;

auLvs – список соединений для DIVA и Assura;

auCdl – список соединений в формате *Circuit Descriptive Language* (язык описания схем) для генерирования списка соединений для Dracula или экспорта в другие симуляторы;

ivpcell – идентифицирующий устройство символ, используемый в экстрагированной топологии при верификации в DIVA или Assura;

layout – представление топологической ячейки (Fixed cell, pcell) в Virtuoso Layout Editor.

представления abstract содержат границы блока и расположение пинов.

lef (*Library Exchange Format*) – для проведения автоматического размещения и трассировки цифровых блоков. Цифровые библиотеки содержат большое количество элементов, различающихся функционально (инверторы, буферы, элементы логики, триггеры и т.д.), и по физическим параметрам (элементы с повышенным быстродействием или с пониженной потребляющей мощностью).

3.4. Элементы библиотеки PDK

Для удобства использования в иерархической структуре схемотехнического и топологического представлений микросхемы большинство простейших ячеек, таких как транзисторы, резисторы, конденсаторы, параметризуются. Это означает, что есть возможность задавать вручную их физические размеры: длину и ширину канала; расстояние контактных областей к стоку или истоку до затвора; количество затворов и т.д. Библиотеки PDK содержат наборы элементов, для каждого из которых в PDK соответствуют символы, модели, правила проектирования и технологические файлы.

1. **МОП-транзистор**, четырехтерминальное представление которого (D – сток, G – затвор, S – исток, B – подложка) имеет следующие виды: symbol, spectre, hspiceD, auLvs, auCdl, ivpcell, layout. Геометрические размеры, а также параметры транзисторов варьируются. Для моделирования и верификации они содержат следующие параметры:

- Model Name – имя модели в Spectre;
- Multiplier – количество параллельно соединенных МОП-транзисторов;
- Length (M) – длина транзистора;
- Total Width (M) – суммарная ширина транзистора;
- Finger Width – ширина единичного сегмента транзистора;
- Fingers – количество сегментов транзистора;

- Gate Connection – соединение затворов нескольких сегментов транзистора;
- S/D Connection – соединение истоков/стоков нескольких сегментов транзистора;
- S/D Metal Width – ширина металлизации для соединения стоков и истоков;
- Switch S/D – удаление контактов к области диффузии истока/стока;

2. Набор **резисторов** содержит три типа: диффузионные, изолированные и металлические. Диффузионные p^+- и n^+-резисторы являются трехтерминальными (PLUS, MINUS, B), изолированные поликремневые и металлические являются двухтерминальными (PLUS, MINUS). Устройства имеют следующие представления: symbol, schematic, auLvs, auCdl, ivpcell, layout. При проектировании варьируются как геометрические размеры: номинал; количество сегментов (последовательно или параллельно соединенных), а также переходное сопротивление. CDF-параметры резисторов:

- Model Name – имя модели для Spectre;
- Segments – количество параллельно или последовательно соединенных резистивных сегментов;
- Segment Connection – количество подключений последовательных или параллельных резистивных сегментов;
- Calculated Parameter – рассчитываемые параметры (сопротивление, ширина, длина);
- Resistance – полное сопротивление, равное сумме сопротивлений всех сегментов;
- Segment Width – ширина резистивного сегмента;
- Segment Length – длина резистивного сегмента;
- Contact Rows/Columns – количество рядов/столбцов контактов;
- Sheet Resistivity – удельное сопротивление;
- Contact Resistance – контактное сопротивления резисторов;
- Temperature Coefficient 1, 2 – температурные коэффициенты.

3. **Конденсаторы** трехтерминальные (G, D/S, B) – symbol, spectre, hspiceD, auLvs, auCdl, ivpcell, layout. CDF-параметры:

- Model Name – имя модели в программе Spectre;
- Multiplier – количество параллельно соединенных конденсаторов;

- Calculated Parameter – рассчитываемые параметры (емкость, длина, ширина);
- Capacitance – полная емкость;
- Length (M) – длина;
- Total Width (M) – полная ширина с учетом всех сегментов;
- Finger Width – ширина единичного сегмента;
- Fingers – количество сегментов в топологии;

4. **Диоды** (PLUS, MINUS) – symbol, spectre, hspiceD, auLvs, auCdl, ivpcell, layout. Все диоды в PDK двухтерминальные, варьируются только длина и ширина. Модели представлены в виде списка соединений как предопределенные устройства. CDF-параметры:

- Model name – имя модели в программе Spectre;
- Calculated Parameter – рассчитываемые параметры (площадь, длина, ширина, периметр);
- Multiplier – количество параллельно включенных диодов.

5. **Биполярные транзисторы**, трехтерминальные (C, B, E) – symbol, spectre, hspiceD, auLvs, auCdl, ivpcell. Все биполярные транзисторы в PDK трехтерминальные. Допускается применение только устройств с фиксированными размерами. Площадь в основном определяется по области эмиттера. CDF- параметры:

- Model name – имя модели в программе Spectre;
- Device Area Emitter – площадь устройства;
- Emitter – ширина эмиттера;
- Multiplier – количество параллельно включенных транзисторов.

3.5. Состав PDK

В PDK предопределены типовые наборы топологических слоев, поддерживаемые средой проектирования и технологиями. Для 90 нм БиКМОП-процесса топологические слои GPDK090 и их семантика приведены в таблице 3.1. Слои делятся на технологические, то есть участвующие в создании фотошаблонов (слои металлов, диффузионные слои и т.д.), и информационные (для распознавания элементов, различные метки, обозначения границ блоков и т.д.).

Для каждого из устройств в PDK сформирован набор слоев. Их подробное описание дано в правилах проектирования и в описании устройств. К каждому устройству приводятся условия их эксплуатации, разброс параметров.

Т а б л и ц а 3.1

Примеры топологических слоев GPDK090

№ GDS	GDS тип	CDS имя	CDS семантика	Описание в терминах PDK
1	0	Oxide	Drawing	Oxide
2	0	Nwell	Drawing	Nwell
3	0	Poly	Drawing	Poly
4	0	Nimp	Drawing	N+ implant
5	0	Pimp	Drawing	P+ implant
72	0	SiProt	Drawing	Salicide Blocking
19	0	Nburied	Drawing	N buried
6	0	Cont	Drawing	Contact
7	0	Metal1	Drawing	Metal1
	1		Pin	
	2		Slot	
	3		Label	
8	0	Via1	Drawing	Via1
9	0	Metal2	Drawing	Metal2
	1		Pin	
	…		…	
…	…	…	…	…
41	0	Via8	Drawing	Via8
42	0	Metal9	Drawing	Metal9
	…		…	
22	0	DIOdummy	Drawing	Recognition layer for diodes
21	0	PNPdummy	Drawing	Recognition layer for pnp
20	0	NPNdummy	Drawing	Recognition layer for npn
13	0	Resdum	Drawing	Recognition layer for resistor
36	0	Bondpad	Drawing	Recognition layer for bondpad
14	0	CapMetal	Drawing	Recognition layer for moscap
75–83	0	M1Resdum–M9Resdum	Drawing	Recognition layer for metal res
74	0	ESDdummy	Drawing	Recognition layer for esd

3.6. Вопросы для самопроверки

1. Что такое PDK и какую информацию он в себе содержит?
2. Назовите функции PDK.
3. Какие типы элементов представлены в PDK?
4. Какие представления элементов Вам известны?
5. Что такое параметризированная ячейка?
6. Какие параметры доступны для изменения в элементах?
7. Приведите примеры технологических и информационных слоев.

4. ПРАВИЛА ПРОЕКТИРОВАНИЯ И ФИЗИЧЕСКАЯ ВЕРИФИКАЦИЯ

4.1. Введение

Для современных субмикронных технологий стоимость производства комплекта фотошаблонов сравнима с затратами на проектирование интегральной схемы, поэтому задача получения работоспособного кристалла с первой попытки чрезвычайно актуальна. Один из основных инструментов, направленных на решение этой задачи – системы физической верификации. Сегодня круг задач подобных систем стал значительно шире. Если несколько лет назад речь шла, в основном, о контроле правил проектирования (DRC) и соответствии топологических данных исходной электрической схеме (LVS), то сегодня необходимо уметь моделировать паразитные эффекты в полупроводниковых структурах, проводить экстракцию параметров с высокой точностью (xRC), решать задачи коррекции топологии с учетом разрешающей способности процесса субмикронной фотолитографии (OPC), использовать новые методы подготовки масок, задействовать специальные инструменты подготовки проекта к производству (DFM), ориентированные на учет особенностей конкретного технологического процесса.

Из данной главы вы узнаете:

- ✓ *Что в себя включает физическая верификация ИМС*
- ✓ *Как развивались правила проектирования*
- ✓ *Как выполнить логические операции над слоями*
- ✓ *Как DRC проверяет топологию*
- ✓ *Зачем LVS создает два нетлиста*
- ✓ *Какие типы экстракций существуют*
- ✓ *Особенности проверок ИМС, выполненных по субмикронным технологиям*

4.2. Примеры правил проектирования

Для большинства слоев, содержащихся в PDK, есть свои правила проектирования, например, минимальная и максимальная ширина, минимальное расстояние между элементами одного слоя до элементов другого слоя. Некоторые примеры правил и их эволюция по мере развития технологий представлены в таблицах 4.1, 4.2 и поясняются рисунком 4.1.

Основными геометрическими терминами в PDK являются (рис. 4.2):

Spacing – расстояние между объектами;

Width – ширина объекта;

Overlap – насколько один объект перекрывает другой;

Enclosure – расстояние от внутренней границы одного многоугольника до внешней границы другого многоугольника;

Area – площадь объекта;

Ratio – отношение площадей объектов.

Обычно названия конкретных правил проектирования являются акронимами того что проверяется и для какого слоя. Например, **S1M1** – расстояние между двумя многоугольниками первого металла, **A1DF –** минимальная площадь диффузии.

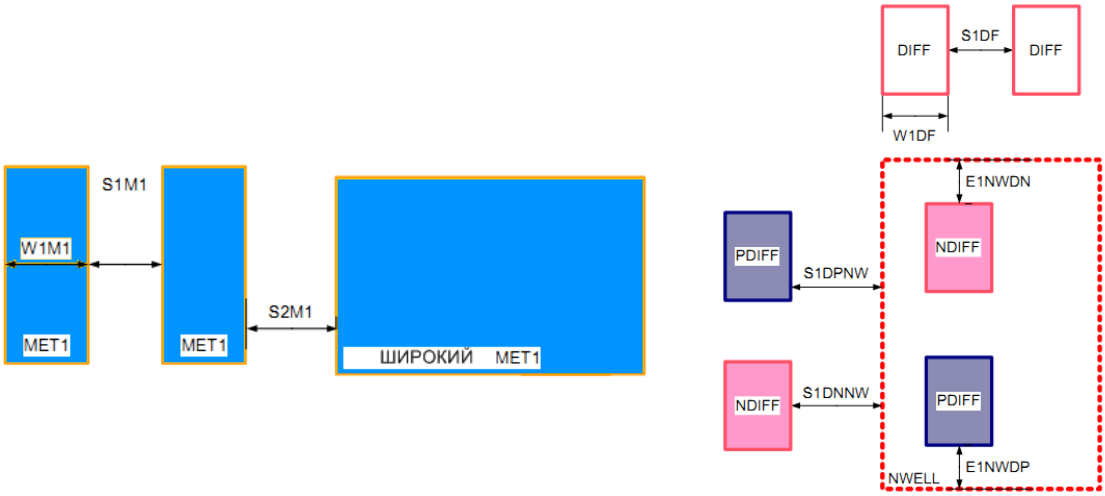


Рис. 4.1. Пояснение к таблице 4.1

Таблица 4.1

Примеры правил проектирования для первого металла и диффузии для технологий с различными топологическими нормами

Пример названия	Описание	1 мкм	0.6 мкм	0.35 мкм	180 нм	90 нм	40 нм	Ед. изм.
W1M1	Минимальная ширина MET1	1.1	0.9	0.5	0.23	0.12	0.07	мкм
	Максимальная ширина MET1					12.0	4.5	мкм
S1M1	Минимальное расстояние между MET1	1.5	0.8	0.45	0.23	0.12	0.07	мкм
S2M1	Минимальное расстояние MET1 до широкого MET1 (шириной >10 мкм или 6мкм)	2.0	1.3	0.8	0.6			мкм
	Минимальное расстояние между MET1 и MET1 (в зависимости от ширины)					0.17, 0.5, 1.5	0.08, 0.12, 0.14, 0.21, 0.5	мкм
S4M1	Минимальное расстояние между MET1 (в зависимости от разности потенциалов)				0.38, 0.6, 1.0			мкм
A1M1	Минимальная площадь MET1	5.76	1.44		0.202	0.058		$мкм^2$
R1M1	Минимальное отношение площади MET1 к площади кристалла	20.0	30.0	30.0	30.0			%
W1DF	Минимальная ширина DIFF	4.0	0.6	0.5	0.22	0.11	0.06	мкм
	Минимальная ширина DIFF при изгибе 45°					0.18	0.17	
S1DF	Минимальное расстояние между DIFF	3.0	1.2	0.6	0.28	0.14	0.08	мкм
S1DNNW	Минимальное расстояние n+ DIFF до NWELL		1.8	1.2	0.43			мкм
S1DPNW	Минимальное расстояние p+ DIFF до NWELL		0.4	0.2	0.12			мкм
E1NWDN	Минимальная протяженность NWELL за пределами пересечения с n+ DIFF		0.4	0.2	0.12			мкм
E1NWDP	Минимальная протяженность NWELL за пределами пересечения с p+ DIFF		1.8	1.2	0.43			мкм
A1DF	Минимальная площадь DIFF				0.202	0.06	0.035	$мкм^2$

Таблица 4.2

Пример правил проектирования для топового уровня металла в шестиметальной технологической опции

Описание	Значение	Ед. изм.
Минимальная ширина MET6	0.44	мкм
Минимальное расстояние между MET6	0.46	мкм
Минимальное расстояние MET6 до MET6 шириной >10 мкм	0.6	мкм
Минимальная площадь MET6	0.562	мкм^2
Минимальное отношение площади MET6 к площади кристалла	30.0	%
Максимальное отношение площади MET6 к площади затвора к которому он подключен	400.0	-
Максимальное отношение площади MET6 к площади кристалла	65.0	%

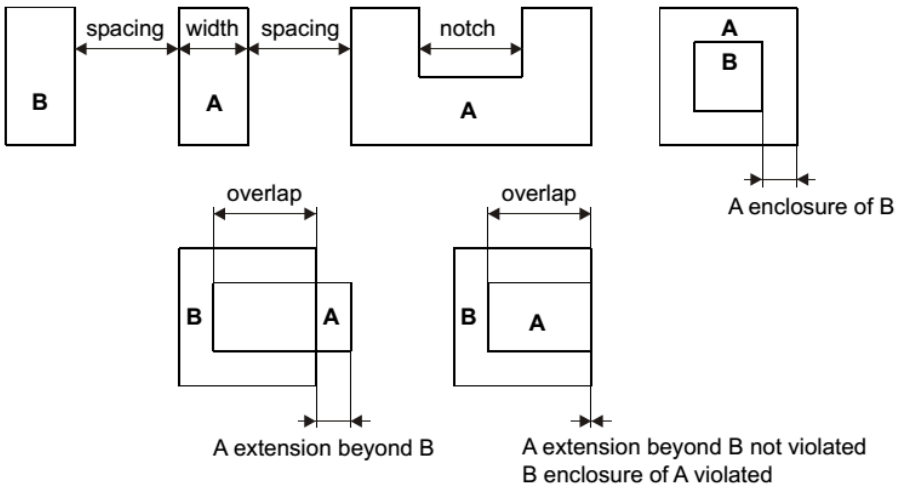


Рис. 4.2. Ключевые проверки на соотвествие правилам проектирования

Из таблиц 4.1 и 4.2 видно, что по мере развития процессов количество правил проектирования неуклонно растет. Если в старых технологиях можно было запомнить все правила проектирования, то в современных технологиях количество правил исчисляется тысячами. Даже самый прилежный человек в мире пропустит что-нибудь из всего множества правил. Достаточно одной неточности, чтобы погубить кристалл. Временной цикл для пластины на фабрике составляет 8 или 12 недель, изготовление обходится

в миллионы рублей, поэтому все хотят быть уверенными, что отправленное на фабрику является корректным.

DRC-программа (*Design Rule Check*) знает всё о конкретном техпроцессе. После её запуска она старательно проверит всё, что нарисовано. Если правила проектирования написаны хорошо, DRC найдет даже малейшие ошибки в топологии. Тогда можно гарантировать, что то, что заложено в кремний, скорее всего, будет функционировать.

DRC-программа помещает в топологию маркеры, которые подсвечивают местоположение ошибок. Исправить все ошибки с первой попытки удается редко, т.к. в процессе исправления ошибок можно добавить новые в другом месте. Исправление DRC-ошибок – это итерационный процесс: запускается DRC, фиксируется ошибки, запускается DRC, фиксируется ошибки, запускается DRC… и работа продолжается по кругу, пока ошибок не останется.

Большая плотность компонентов, их малые размеры, большое количество правил проектирования требуют значительных компьютерных ресурсов. Поэтому развиваются новые подходы к проверкам. Благодаря методам распознавания неявных иерархических структур и учета повторяемости блоков, процесс верификации в современных программах-верификаторах выполняется для повторяющихся ячеек однократно, а далее используются уже готовые результаты этой проверки. Применение подобных методов значительно увеличивает производительность DRC-процедуры. Еще в несколько раз время проверки можно сократить благодаря возможности распараллеливания процессов.

В основе DRC-проверки лежат два этапа: первый заключается в поиске элементов топологии и специфических областей, второй – непосредственная проверка соответствию правилам проектирования. Распознавание устройств в топологии связано с логическими операциями над слоями. Рассмотрим их немного подробнее.

4.3. Логические операции над слоями

Вы знакомы с простыми логическими операциями AND или OR, и знаете, что *true* AND *true* дает *true*. Такие же операции можно выполнять над двухмерными топологическими слоями. Если есть два слоя, то в результате логических операций над ними получится третий слой.

4.3.1. AND-функция

Если два объекта перекрывают друг друга, то результат будет ненулевой. Например, есть два многоугольника *A* и *B*. Перекрытие представляет собой область, где оба многоугольника присутствуют (рис. 4.3).

AND-функция хорошо подходит для обнаружения КМОП-транзисторов. Например, КМОП-транзисторы содержат слой поликремния и активный слой. Там, где поликремний пересекает активную область, и расположен транзистор (рис. 4.4).

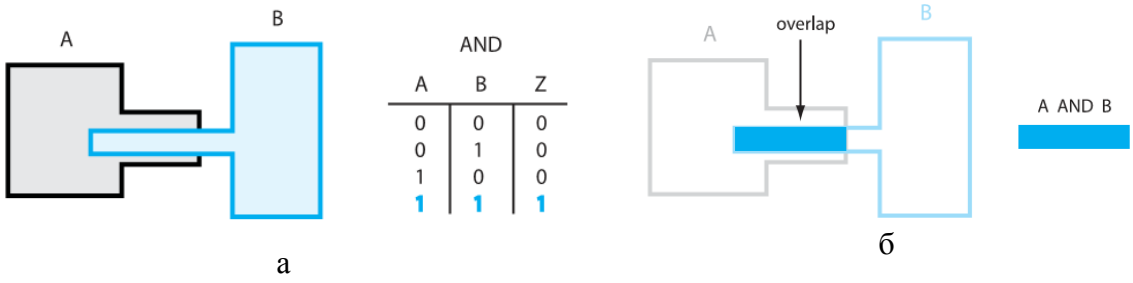


Рис. 4.3. AND-функция:
а – таблица истинности AND-функции и два исходных многоугольника A и B;
б – результат A AND B

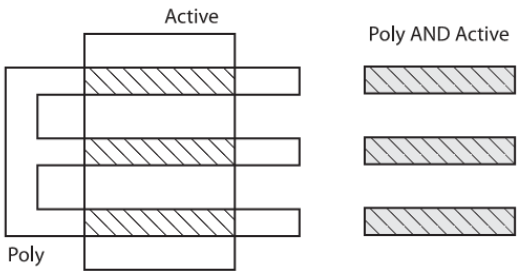


Рис. 4.4. Поиск затворов транзисторов с помощью AND функции

Выходные данные логической функции можно сохранить во временный слой, и воспользоваться им для проведения других логических операций. Слой, который создан при выполнении логический операции *poly* AND *active* – это не технологический слой. Это только временный файл, с которым можно работать дальше. Например, назовем новый слой TMP1, строка кода будет выглядеть так:

TMP1 = POLY AND ACTIVE

Это типичная строчка, которую можно написать в контрольном файле. Выполним над TMP1 следующую операцию:

TMP2 = TMP1 AND NWELL

TMP1 содержит все КМОП-транзисторы, поэтому в результате операции *TMP1 AND NWELL*, будут распознаны все *P*-МОП-транзисторы. Поэтому TMP2 слой будет содержать положения затворов всех *P*-МОП-транзисторов. После окончания верификации можно будет удалить TMP1 и TMP2.

Выделив *P*-МОП-транзисторы, можно проверить правила, применимые именно к ним. Основная часть правил направлена на поиск положений устройств за счет предварительной работы. Можно сделать так много логических последовательных операций, сколько потребуется, чтобы распознать все компоненты. После обнаружения устройств проводится проверка соответствующими для них правилами.

4.3.2. OR-функция

Следующей широко используемой функцией является OR-функция. Выход не нулевой, если на входе есть хоть что-то. Выход OR-функции выглядит как более крупный многоугольник, чем исходные (рис. 4.5). Функция OR сливает два полигона в один.

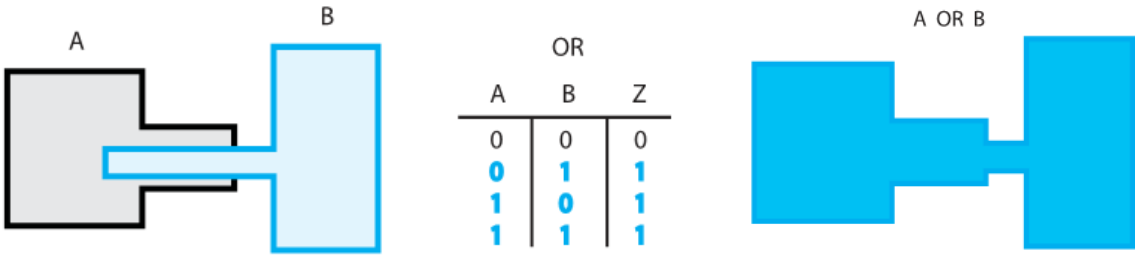


Рис. 4.5. OR-функция:
а – таблица истинности OR-функции исходные многоугольники A и B;
б – результат A OR B

Ниже представлен код, который определяет слой TMP3 как объединение реального слоя (PPLUS – p^+-поликремний) с ранее определенным временным слоем TMP2:

TMP3 = TMP2 OR PPLUS

4.3.3. NOT-функция

Иногда возникают проблемы с NOT-функцией. NOT-функция, в частности в DRC, может быть описана как AND NOT-функция. Это пояснение помогает проще понять, что происходит. Рассмотрим пример.

TMP4 = A NOT B тоже самое, что *TMP4 = A AND NOT B*

Посмотрите на многоугольники на рисунке 4.6а. Где расположено пересечение многоугольника A и *NOT B* (т.е. с внешней стороны многоугольника)? Пересечение *A AND NOT B* показано на рис. 4.6б. NOT-функция чувствительна к последовательности. Например, результат *B NOT A* будет другим (рис. 4.6.б)

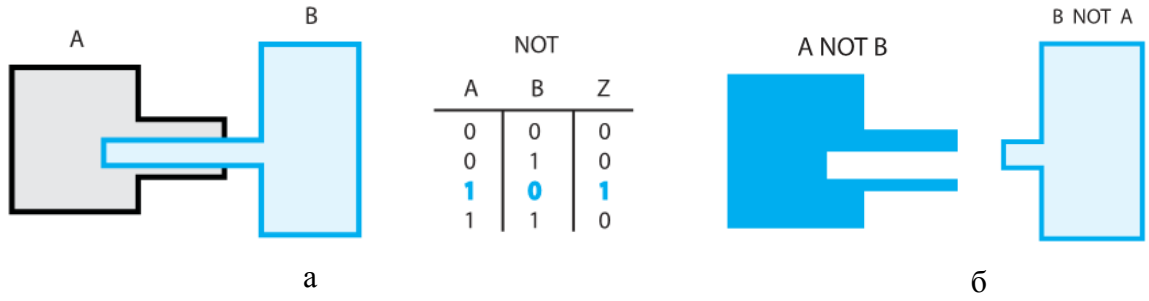


Рис. 4.6. NOT-функция:
а – таблица истинности NOT-функции и два исходных многоугольника A и B;
б – результат A NOT B и B NOT A

NOT функция полезна, например, при разделении резисторов с разным уровнем легирования. Если есть слой, содержащий все поликремниевые резисторы, то операция с NOT специального слоя легирования (рис. 4.7) позволяет отбросить резисторы с другим уровнем легирования и оставит только обычные резисторы:

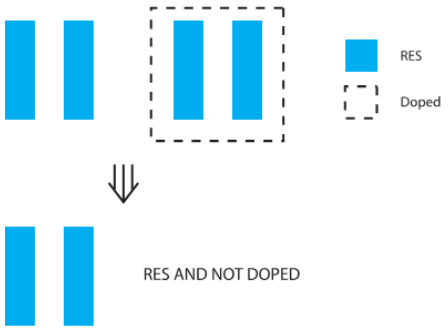


Рис. 4.7. Пример использование NOT-функции
для выделения поликремниевых резисторов с разным уровнем легрования

Таким образом, логические AND-, OR- и NOT-функции очень полезны для обнаружения маленьких индивидуальных фрагментов схемы. Можно записать все команды вместе, создавая временный слой, который содержит конкретное специфическое устройство.

До сих пор, командные строки DRC не проверяли устройства на соответствие правилам, определялись только слои, содержащие устройства. После того они найдены, можно выполнить проверки.

4.4. Командные строки правил проектирования

Для всех слоев, из которых состоят распознанные устройства, обычно проверяются несколько правил проектирования. Первой проверкой, как правило, является расстояние между объектами. В частности, можно проверить является ли расстояние между металлами правильным, т.е. не слишком ли они близко расположены друг к другу. Данную проверку можно проводить как над реальными слоями, так и над произвольными. Рассмотрим пример EXTERNAL-проверки:

DISPLAY CHECK1 = EXTERNAL M1>= 2 um (рис. 4.8а)

Эта строка кода проверяет, чтобы расстояние между внешними границами многоугольников расположенных напротив друг друга было больше или равно 2 мкм. В этом примере, слово *EXTERNAL* распознается программой-верификатором как предопределенная команда. Программа поместит маркерный слой, который светится, чтобы легко определить место расположения ошибки.

Рис. 4.8. Проверка геометрических параметров:
а – расстояние между двумя прямоугольниками первого металла; б – ширина объекта

Следующей общей проверкой является проверка ширины. Ниже представлена строка кода, проверяющая ширину первого металла. Любой объект шириной менее 2 мкм будет подсвечен. Ошибки являются выходным слоем, названным CHECK2:

DISPLAY CHECK2 = WIDTH M1<2um (рис. 4.8б)

Обычно командные строки пишутся в виде текстовых файлов. Программное обеспечение запускает последовательно одну операцию за другой. Строки, определяющие временные слои через логические операции, находятся в том же DRC-файле, что и команды правил. Временные слои определяются первыми. Затем следуют строки проверок правил проектирования.

Каждая программа имеет свой собственный синтаксис, свой собственный язык. Каждая проверка будет иметь список опций. Чем больше слоев и правил проверяется, тем больше контрольный файл. Он может содержать тысячи и тысячи строк.

4.5. Программы проверки DRC

4.5.1. Diva DRC

Запуск приложения Diva осуществляется из меню Verify/DRC топологического редактора Virtuoso. Для проверки соответствия правилам проектирования необходимо указать (рис. 4.9):

Checking Method – метод проверки: flat – проверка топологии без учета иерархии; hierarchical – иерархическая проверка;

Join Nets With Same Name – виртуальное соединение шин с одинаковыми именами;

Rules File – название и место расположение файла правил.

После завершения проверки проектных норм в окне icfb будет выведена информация о найденных ошибках в формате, показанном на рисунке 4.10.

Diva DRC информирует о числе обнаруженных ошибок для каждого из нарушенных правил и об общем количестве ошибок. При нахождении ошибки программа автоматически размещает в топологии маркер на месте обнаружения ошибки. Для просмотра ошибок необходимо выбрать в меню «Verify/Marker/Find…». При выполнении команды откроется окно Find Marker, показанное на рисунке 4.11.

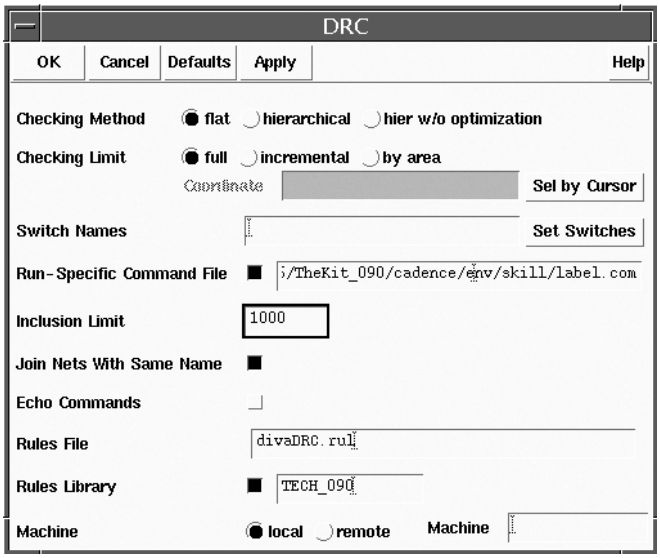


Рис. 4.9. Окно программы Diva DRC

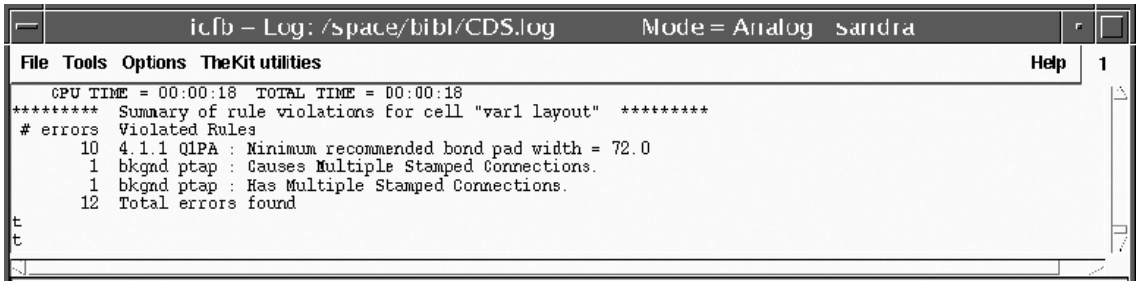


Рис. 4.10. Сведения об обнаруженных ошибках DRC

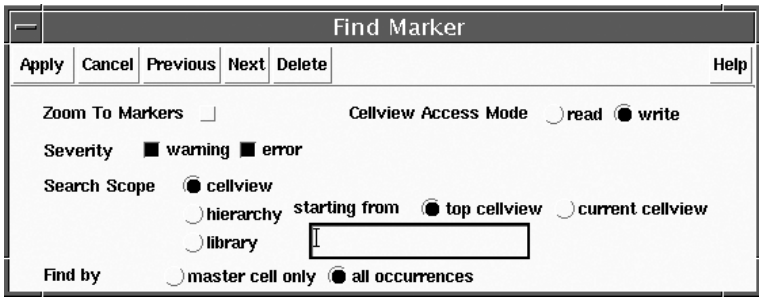


Рис. 4.11. Окно поиска ошибок Find Marker

Основные установки, необходимые для просмотра и исправления ошибок:

Zoom To Marker – увеличение места ошибки до размера экрана;

Severity – выбор варианта просмотра: warning – просмотр предупреждений; error – просмотр ошибок;

Search Scope – границы поиска ошибок: cellview – в ячейке; hierarchy – иерархический поиск; library – в указанной библиотеке. Для типов cellview и hierarchy доступны дополнительные настройки: начать поиск с top cellview – верхней ячейки или current cellview –текущей ячейки;

Find by – поиск в: master cell only – только в верхней ячейке; all occurrences – во всех ячейках.

После того как все настройки определены при нажатии на кнопку «Next» будет осуществлен переход к месту следующей ошибки, при нажатии на кнопку «Previous» будет осуществлен переход к предыдущей ошибке. При нахождении места первой ошибки открывается окно marker text, в котором будут указаны в текстовом виде: location: (библиотека, название ячейки, тип представления) – место нахождения ошибки; reason: нарушенное правило.

4.5.2. Assura DRC

Запуск приложения Assura DRC, предназначенного для проверки проектных норм, осуществляется из меню «Assura/Run DRC» редактора топологии Virtuoso. В результате выполнения команды на экран выводится окно, показанное на рис. 4.12.

Для осуществления проверки необходимо задать следующие настройки программы:

1) место расположения проверяемого файла: а) Library – библиотека; б) Cell – название ячейки проекта; в) View – представление (обычно Layout);

2) рабочая среда: а) Run Name – имя процесса; б) Run Directory – рабочая директория, где будет сохраняться текущий выполняемый процесс (по умолчанию Assura DRC);

3) параметры технологии проектирования: а) технологический процесс (Technology); б) место расположения и название управляющего файла проверки.

После выполнения проверки DRC Assura выведет результаты проверки в окне Error Layer Window – ELW (рис. 4.13).

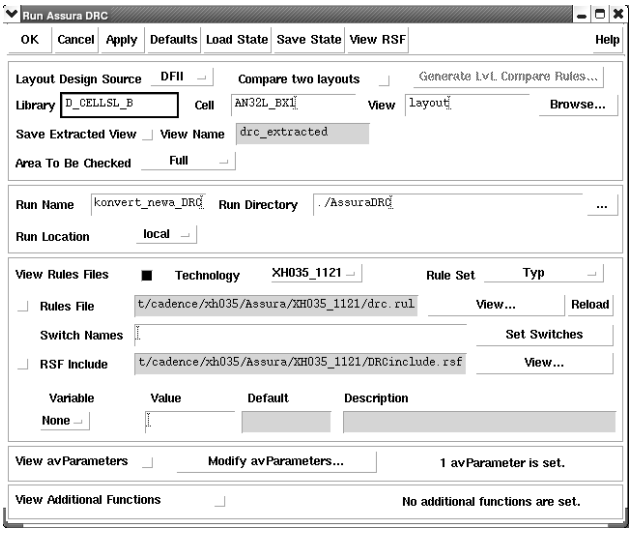


Рис. 4.12. Рабочее окно программы Assura DRC

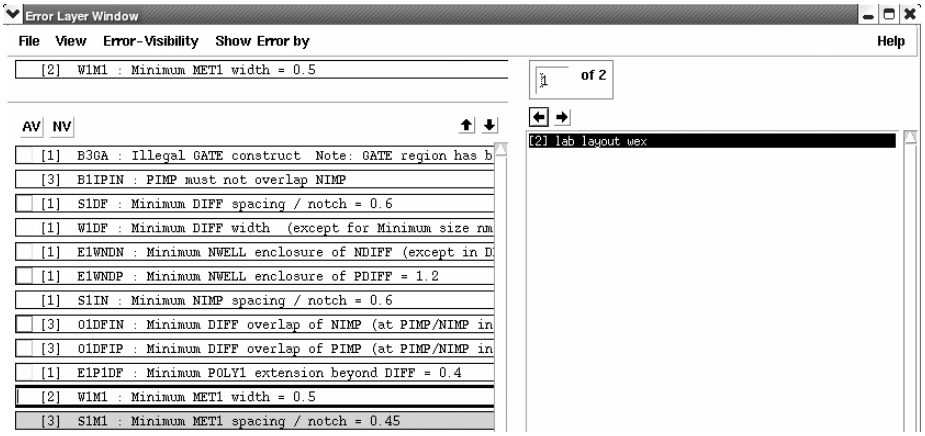


Рис. 4.13. Окно поиска исправления ошибок Error Layer Window

В левой части окна ELW перечислены правила проектирования, по которым обнаружены ошибки. В каждой строке сначала указывается цвет, которым будет подсвечена ошибка данного правила, затем в квадратных скобках указано число ошибок и название правила. Обычно также указывается необходимое минимальное/максимальное численное значение требований в «мкм». Вверху слева указывается текущее выбранное правило и его название. Справа в поле для текущего правила указывается топологическая ячейка, в которой эта ошибка найдена. С помощью стрелок вправо/влево осуществляется переход к следующей/предыдущей ошибке, которая будет подсвечена в топологии в виде цветного многоугольника.

Дополнительные возможности: кнопки «AV» и «NV» показывают/скрывают все подсвеченные ошибки.

4.5.3. Calibre DRC

Запуск приложения Calibre DRC осуществляется из меню Calibre/Run DRC окна редактора топологии Virtuoso (рис. 4.14). Перед началом проверки необходимо задать название и место расположения файла проверки calibre.drc, а также место размещения и название рабочего каталога Run Directory. Во вкладке Input необходимо задать: тип проверки (иерархический или одноуровневый); имя входного GDS-файла; название ячейки верхнего уровня. Во вкладке Outputs требуется определить: название базы данных с результатами проверки; формат базы данных; имя выходного файла-отчета о проверке.

В меню Setup/Select Checks… осуществляется выбор правил/групп правил проектирования, которые необходимо проверить (рис. 4.15). После окончания проверки, появится окно Calibre DRC RVE (рис. 4.16), в котором отобразятся результаты проверки правил проектирования.

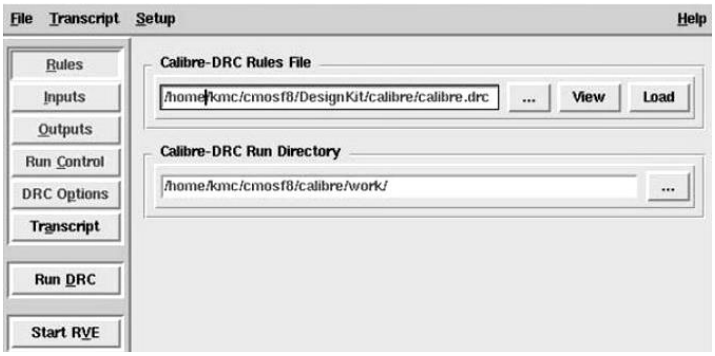


Рис. 4.14. Окно проверки правил проектирования Calibre DRC

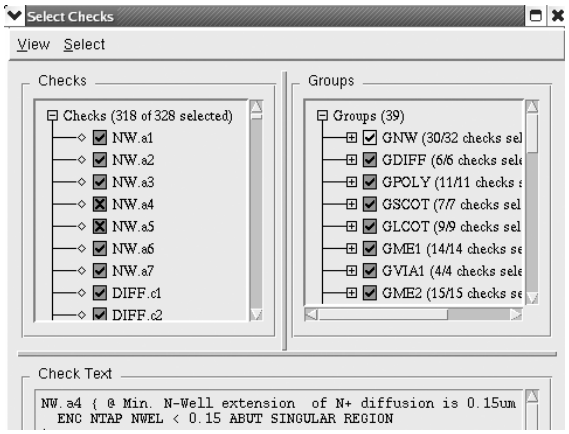


Рис. 4.15. Окно выбора правил проектирования

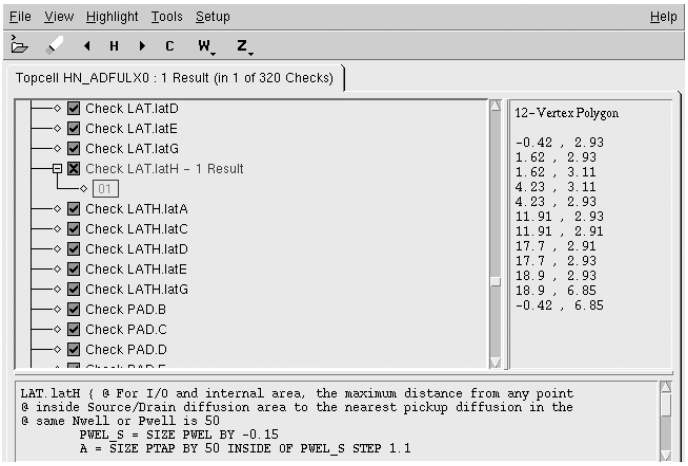


Рис. 4.16. Окно поиска и просмотра ошибок Callibre DRC RVE

Правила, не прошедшие проверку, будут выделены маркерами. Для просмотра ошибок по любому из этих правил необходимо раскрыть список и кликнуть по одной из ошибок. При этом в топологии будет подсвечена область ошибки, в правой части окна Calibre DRC RVE будет отображена информация о координатах положения ошибки, а в нижней части будет выведен отчет, содержащий текстовую информацию об ошибке и правило, по которому осуществлялась проверка.

4.6. Проверка на соответствие электрической схеме и экстракция паразитных элементов

DRC всего лишь первый уровень проверки. Даже если топология не содержит DRC ошибок, это не означает, что все подключения выполнены корректно. Например, может оказаться, что разные сети закорочены или наоборот – не подключены. Второе программное обеспечение, входящее в состав стандартной верификации называется LVS (*Layout Versus Schematic).* LVS отвечает за восстановление принципиальной электрической схемы из топологического представления и сравнение полученных результатов с исходным описанием электрической схемы. При восстановлении электрической схемы решаются задачи идентификации компонентов электрической схемы и соединений, определения их параметров на основе тех же команд, что использует DRC. По-прежнему используются логические операции над слоями для поиска устройств. Но расширения к правилам проектирования позволяют LVS не только обнаруживать элементы, но и на основании геометрических данных рассчитать реальные параметры компонентов и сравнить с параметрами, использованными электрической схеме. Таким об-

разом, процесс LVS-проверки состоит из двух этапов. Первая часть процесса – это извлечение информации об устройствах из топологии или извлечение соединений (*connectivity extraction*). При извлечении элементов создается нетлист (*netlist*), который обычно представляет собой текстовый файл. Каждая строчка определяет один найденный компонент и информацию о других компонентах, к которым он подключен. Вторая часть LVS-процесса – сравнение. Большинство современных программ используют два нетлиста. Программа извлекает нетлист устройств, обнаруженных в топологии, затем генерирует нетлист из схематика, и затем сравнивает два нетлиста.

Рассмотрим простой SPICE-нетлист, сгенерированный LVS-программой после извлечения компонентов.

```
R1   A   B   10K       PPLUS
Q1   A   D   E   F     NPN       A=5
C1   E   J   5P        MOSCAP
```

Читаем нетлист:

Первая строчка: обнаружен резистор, который называется R1. Он подключен между узлами A и B. Номинал резистора 10 000 Ом, тип резистора P+.

Вторая строчка: следующее найденное устройство – транзистор Q1. Он подключен между точками A, D и E. Четвертый вывод, который обычно является подложкой, подключен к F. Это *N–P–N*-транзистор площадью 5 мкм.

Третья строчка: относится к емкости, подключенной между узлами E и J. Величина емкости 5 пФ, тип емкости MOSCAP.

Выходной нетлист может быть очень длинным, так как могут быть распознаны тысячи устройств. Нетлист формирует текстовую версию того как устройства подключены в топологии. Попробуем транслировать в схематик рассмотренный нетлист (рис. 4.17).

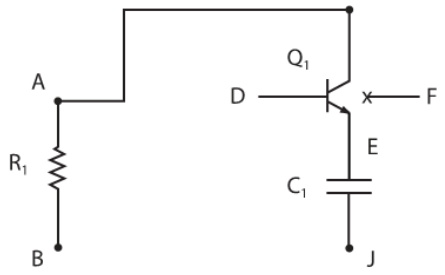


Рис. 4.17. Схематик, полученный из нетлиста, проедставленного выше

LVS может работать как с иерархическими, так и с одноуровневыми проектами. Также есть средства распознавания повторяющихся логических элементов и элементов памяти, что позволяет значительно повысить производительность процесса верификации. Средства визуализации позволяют анализировать результаты верификации в интерактивном режиме.

Модуль LVS непосредственно связан с модулем извлечения паразитных элементов PEX (*parasitic extraction*) и xRC (*extraction of resistors and capacity*), без учета которых проектирование современных схем уже практически невозможно. Модуль xRC обеспечивает экстракцию параметров соединений на уровне транзисторов, вентильном уровне, а также на уровне иерархических блоков, поэтому может быть интегрирован в самые разные маршруты проектирования.

Результаты экстракции сохраняются в специальной базе данных паразитных параметров и используются для различных средств анализа: анализ целостности сигналов; временной анализ; анализ надежности; анализ энергопотребления и системы питания (рис.4.18).

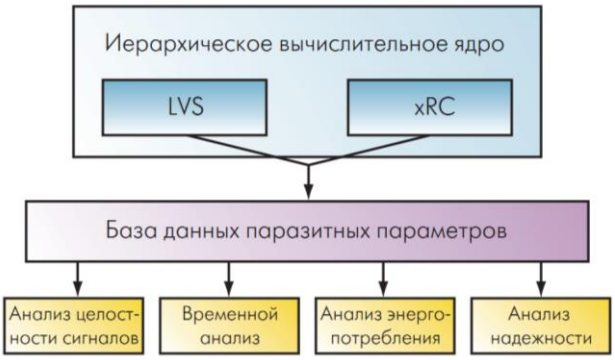


Рис.4.18. Поддержка паразитных параметров для различных типов анализа

Наиболее распространены следующие виды экстракций:

- R – извлечение устройств и паразитных сопротивлений;
- R-C – извлечение устройств, паразитных сопротивлений и паразитных емкостей на подложку, емкости между объектами учитываются путем построения эквивалентных емкостей на подложку;
- R-C-CC – извлечение устройств, паразитных сопротивлений, паразитных емкостей на подложку и емкостей между объектами (*coupling capacitance*);

- C-CC – извлечение устройств и паразитных емкостей;
- noRC – извлечение устройств без паразитных элементов.

Существуют дополнительные виды экстракций, например, L – позволяет проводить экстракцию паразитных индуктивностей.

4.7. Программы проверки LVS

4.7.1. Diva LVS

Для того чтобы осуществить верификацию топологии и электрической схемы в программе Diva, необходимо в меню редактора топологии Virtuoso нажать «Verify/LVS…». В результате выполнения команды появляется окно, показанное на рисунке 4.19.

Основные настройки программы верификации Diva LVS:

Run Directory – рабочая директория процесса верификации;

Create Netlist – создание нетлиста: Schematic – из схемы, далее указывается ячейка схемы; extracted – из извлеченной схемы, далее указывается название экстрагированной схемы;

Rules File – управляющий файл верификации;

Rules Library – технологическая библиотека;

LVS Options – дополнительные опции LVS: Rewiring – комбинирование симметричных сетей схемы; Device Fixing – фиксация приборов схемы; Create Cross Reference – проверка перекрещивающихся сетей, в случае нахождения таковых формируется совет о замене подключения прибора к другой сети вместо текущей; Terminals – учет при верификации расставленных пинов.

Кнопки:

Run – запуск процесса верификации;

Output – отчет верификации;

Error Display – запуск отображения обнаруженных ошибок.

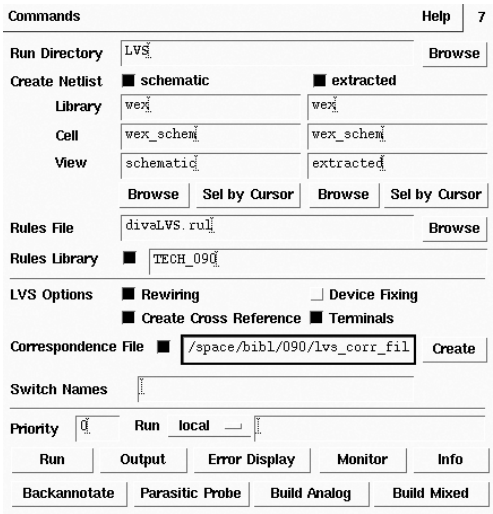


Рис. 4.19. Окно программы верификации Diva LVS

Содержимое окна Probing (рис. 4.20):

Probing Method – метод поиска ошибок: single w/o parasitics – выделение ошибки только в выбранном окне без паразитных элементов; single probe matched – выделение прошедших проверку сетей в выбранном окне; single probe all – выделение всех сетей в выбранном окне; cross probe matched – выделение сопоставленных сетей одновременно в окне схемы и окне извлеченной схемы; single probe unmatched – выделение ошибочных сетей в одном окне;

Клавиши: Add Dev – подсветить прибор; Remove Dev – снять выделение прибора; Add Net – выделить сеть; Remove Net – снять выделение с сети; Add Nets for Dev – выделить сети, подключенные к прибору; Remove Nets for Dev – снять выделение с сетей, подключенных к прибору; Add Devs for Net – выделить приборы, подключенные к сети; Remove Devs for Net – снять выделение с приборов, подключенных к сети; Remove All – снять выделение со всех приборов и сетей.

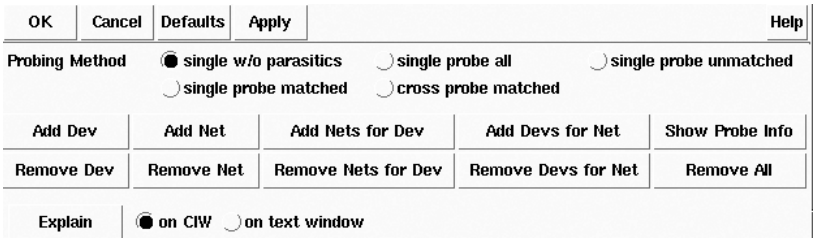


Рис. 4.20. Окно поиска ошибок Probe

Сразу после завершения работы программа Diva LVS выводит отчет о завершении процесса верификации. Поиск ошибок происходит следующим образом. После нажатия соответствующей кнопки следует указать требуемую сеть или прибор, например, в схеме. В случае нахождения соответствия между электрической схемой и экстрактом выделится соответствующая сеть в экстракте. Если сеть не была сопоставлена, то она не будет выделена. Выделяя таким образом входы, выходы, приборы и сети, находят между ними соответствия. Анализируя и сопоставляя схему и топологию, находят ошибки.

Экстракция выполняется с помощью приложения Diva EXT. Запуск приложения выполняется из меню «Verify/Extract» (рис. 4.21). Основные настройки программы-экстрактора:

Extract Method – метод экстракции;

Join Nets With Same Name – соединение сетей с одинаковыми названиями;

Switch Names – переключатель, устанавливается в случаях, когда требуется отключение отдельных правил экстракции;

Rules File – управляющий файл экстракции divaEXT.rul;

Rules Library – технологическая библиотека.

После выполнения экстракции в рабочем каталоге появляется представление ячейки extracted, содержащее топологию с наложенной на нее электрической схемой. Топологические слои в данном файле представлены в net-формате.

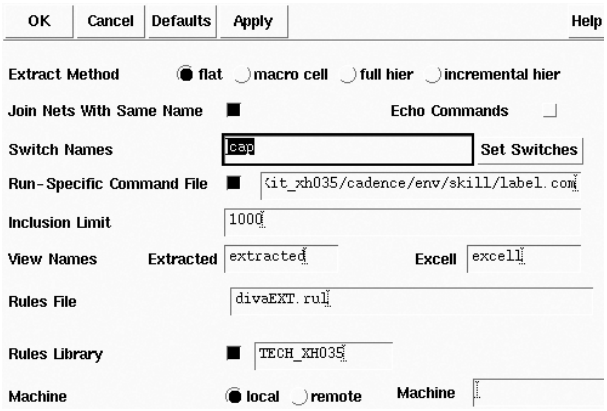


Рис. 4.21. Окно экстракции приложения Diva Extactor

4.7.2. Assura LVS

Запуск приложения верификации Assura LVS выполняется из меню «Assura/Run LVS» редактора топологии Virtuoso. В результате выполнения команды появляется окно (рис. 4.22). Для работы приложения необходимо задать:

- название схемотехнической ячейки и библиотеку ячейки;
- название топологической ячейки и библиотеку ячейки;
- название и путь к файлу извлечения списка электрических связей (*extract.rul*);
- название и путь к файлу сравнения списков электрических связей топологии и схемы (*compare.rul*).

Сразу после запуска процесса верификации программа-экстрактор начинает извлечение электрической схемы из топологии. Экстракция происходит согласно правилам, определенным в файле extract.rul. После извлечения электрической схемы из топологии автоматически запускается процесс верификации. После окончания верификации на экран будет выведен файл отчета. В случае нахождения ошибок на экран будет выведен графический интерфейс отладки LVS Debug, показанный на рис. 4.23.

В правой части окна указаны все ошибочные приборы, сети и терминалы. Для того чтобы посмотреть ошибки необходимо выбрать тип ошибок, которые будут выведены «Rewires», «Nets», «Devices», «Pins», «Parameters», и нажать на клавишу «Open Tool». При этом на экран выводится окно ошибок выбранного типа, например, показанное на рис. 4.24. В верхней строке окна указаны названия ячеек (директорий) схемы и топологии. Строка ниже содержит информацию о расхождении параметров. Нижняя половина окна разбита на две части: информация о приборах топологии и информация о приборах схемы.

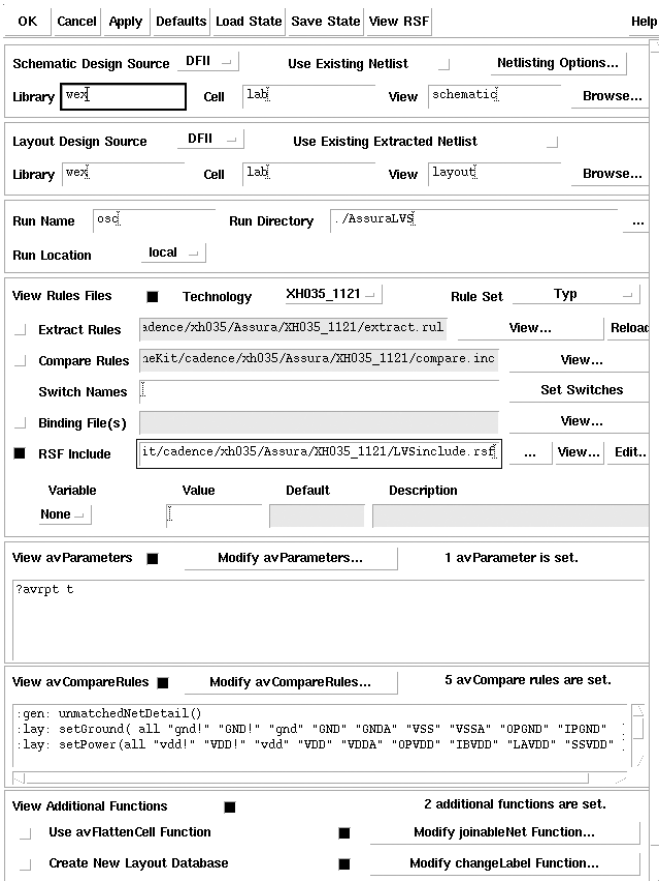


Рис. 4.22. Окно Assura LVS

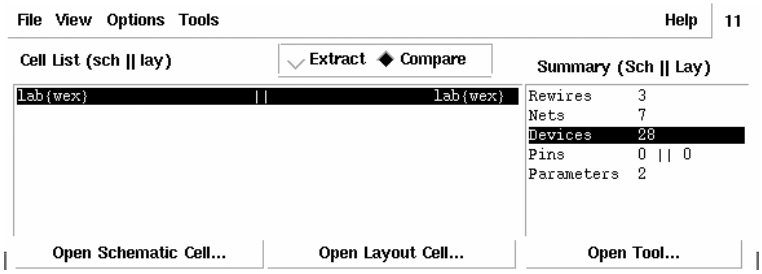


Рис. 4.23. Окно отладки LVS Debug

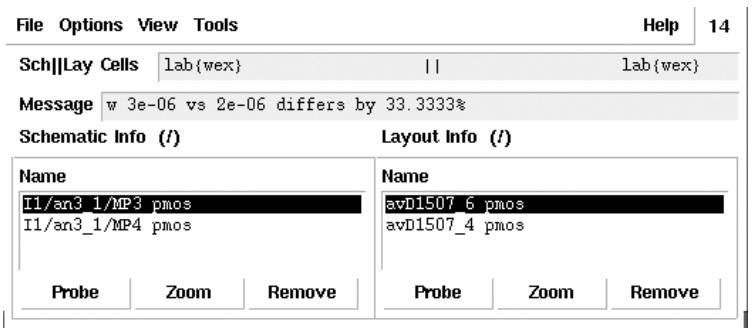


Рис. 4.24. Окно ошибок Parameters Mismatch Tool

4.7.3. Calibre LVS

Запуск приложения верификации Calibre LVS выполняется из меню «Calibre/LVS» окна редактора топологии Virtuoso. После выполнения команды на экран будет выведено окно Calibre LVS (рис. 4.25), содержащее пять вкладок.

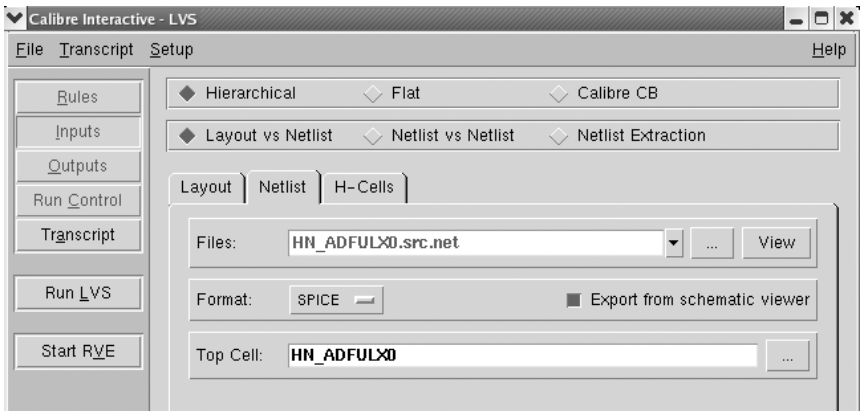


Рис. 4.25. Вкладка Inputs программы Calibre LVS

Настройки вкладки Rules:

Rules – название и место расположения управляющего файла;

Run Directory - рабочий каталог программы.

Настройки вкладки Input:

Тип проверки (иерархическая, одноуровневая);

Варианты сравнения: Layout vs Netlist – топология и нетлист; Netlist vs Netlist – нетлист и нетлист; Netlist Extraction – экстракция нетлиста;

Закладка Layout (Топология): File – имя базы данных топологии; Format – формат базы данных; Export form layout viewer – экспорт из редактора топологии; Top Cell – название ячейки верхнего уровня; Layout Netlist – название нетлиста, извлеченного из топологии;

Закладка Netlist (схема): Files – название файла нетлиста; Format – формат нетлиста; Export from schematic viewer – экспорт из редактора схемы; Top Cell – название ячейки верхнего уровня.

Настройки вкладки Outputs подобны настройкам Calibre DRC.

Просмотр и исправление ошибок после окончания проверки выполняется аналогично DRC-проверке в интерактивной программе Calibre RVE.

4.8. Дополнительные типы проверок для субмикронных технологий

4.8.1. Учет разрешающей способности оборудования (RET)

При переходе к субмикронным нормам геометрические размеры элементов становятся соизмеримы с длиной волны света. Оборудование фотолитографии не может обеспечить точное воспроизведение геометрии элементов. Это приводит к появлению многочисленных дефектов и резкому снижению выхода годных кристаллов. В настоящее время разработан ряд методов повышения разрешающей способности оборудования (оптической коррекции). Дополнительные проверки позволяют:

1. Выполнить контроль специальных правил проектирования, основанных на оценке воспроизводимости отдельных элементов геометрии с учетом краевых эффектов субмикронной фотолитографии;
2. Выполнить оптическую коррекцию геометрии на основе заранее вычисленных табличных правил;
3. Провести моделирование, которое позволяет предсказать, как будут выглядеть геометрические объекты топологии в процессе их "переноса" на кремниевую пластину. Результаты моделирования отображаются на специальных слоях, для которых можно выполнить DRC проверку, имитирующую проверку реального кристалла (*simulated silicon*);
4. Провести коррекцию топологии кристалла, направленную на увеличение выхода годных и повышение надежности технологического процесса. Процесс коррекции заключается во внесении изменений, компенсирующих фотолитографические искажения (рис. 4.26).
5. Выполнить генерацию данных для формирования специальных линий в областях, где необходимо получить четкое изображение. Они позволяют дополнительно сфокусировать луч в процессе фотолитографии и уменьшить эффект аберрации и дисперсии;
6. Создать модели технологического процесса и файлов настройки технологического оборудования.

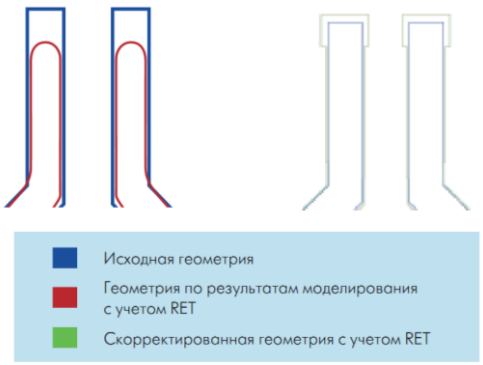


Рис.4.26. Коррекция топологии с учетом RET

4.8.2. Подготовка проекта к производству (DFM)

С уменьшением размеров компонентов интегральных схем проблема обеспечения высокого выхода годных должна быть в поле зрения разработчиков на всех этапах проектирования. Крупные производители интегральных схем предлагают собственные наборы правил проектирования, основанные на производственном опыте (DFM). Правила эти обычно предъявляют более жесткие требования к геометрии элементов и зазорам, минимизируя риск возникновения дефектов в результате возможных отклонений в процессе фотолитографии. На основе учета "производственных" правил можно обнаружить потенциальные места возникновения дефектов и провести коррекцию. Правила позволяют осуществить:

1. Анализ геометрических элементов на удовлетворение правилам DFM с точки зрения ширин и зазоров и формирование статистики распределения нарушений по кристаллу. Это позволяет выделить проблемные области и устранить нарушения в процессе отладки.

2. Анализ геометрических элементов с точки зрения степени их влияния на выход годных и присвоить им соответствующие весовые коэффициенты. Элементы могут быть отсортированы по определенным характеристикам, а результаты анализа представлены либо в виде гистограмм, либо в виде объектов, выделенных непосредственно в топологическом представлении;

3. Анализ одиночных межслойных переходов и их возможные послойные смещения, которые могут привести к ненадежному контакту и повлиять существенным образом на выход годных. В случае необходимости автоматически добавляются дублирующие межслойные переходы, исключающие вероятность отказа.

4.9. Вопросы для самопроверки

1. Что такое физическая верификация ИМС? Для чего она выполняется?
2. Назовите примеры правил проектирования.
3. Какие логические операции над слоями вам известны?
4. Опишите этапы работы DRC-программы.
5. Опишите этапы работы LVS-программы.
6. Что такое нетлист? Зачем проводится моделирование экстрагированных из топологии схем?
7. Какие типы экстракций вам известны?
8. Какие дополнительные проверки необходимы в субмикронных технологиях?

5. БИБЛИОТЕКИ ТОПОЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

5.1. Введение

Все устройства в PDK делятся на активные (транзисторы, диоды) и пассивные (резисторы, конденсаторы).

Из данной главы вы узнаете:

- *Как выглядит топология и сечение пассивных и активных элементов*
- *Какие типы резисторов доступны разработчикам*
- *Зачем в топологии нужны "змейки"*
- *Что такое МОП, МОМ и MIM конденсаторы*
- *Увеличится ли емкость конденсатора в 4 раза, если увеличить его площадь в 4 раза*

5.2. Пассивные элементы

Выбор и количество видов пассивных элементов в интегральных схемах серьезно ограничено. Планарный процесс хорошо освоен для активных приборов. Пассивным элементам уделяется второстепенное значение. Одновременное создание активных и пассивных приборов в рамках одного технологического цикла требует ряда компромиссов, как в части конструирования, так и исполнения. Это сказывается в ограничении размеров и видов пассивных элементов, наличии больших разбросов номиналов элементов. Кроме того, получение элементов возможно лишь в ограниченном диапазоне значений номиналов. Следующим недостатком интегральных пассивных компонентов является большая занимаемая площадь, что определяет наличие больших паразитной емкостей. Тем не менее, пассивные элементы широко используются в аналоговых ИМС.

5.2.1. Резисторы

В зависимости от используемого технологического слоя резисторы бывают диффузионные, поликремниевые и металлические. Изменением типа легирования и концентрации примеси регулируется удельное сопротивление, температурная зависимость, разброс по номиналам. Диапазон величин сопротивлений резисторов составляет от мОм/□ для металлических, десятков Ом/□ для сильнолегированных диффузионных и

поликремниевых резисторов с силицидом и нескольких кОм/□ (*N*-карман и специальные слои поликремния).

Топология простейшего резистора представляет собой прямоугольную область в резистивном слое (рис. 5.1). Для распознавания программами-верификаторами резисторов, они содержат специальные информационные слои, например Resdum, ResDef (*resistor definition*).

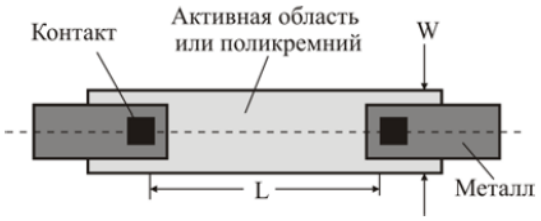
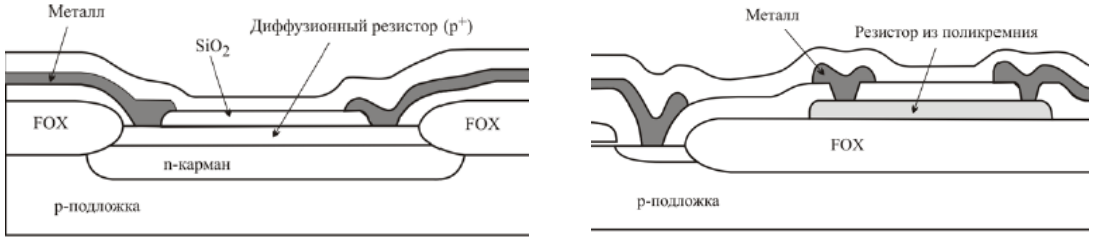


Рис. 5.1. Топологическое представление резистора

Поперечное сечение диффузионного и поликремниевого резисторов для технологии LOCOS представлены на рисунке 5.2.



а б

Рис. 5.2. Резисторы: а – диффузионный (LOCOS); б – поликремниевый (LOCOS)

Общее сопротивление можно выразить соотношением: $R = \frac{\rho L}{S} = \frac{\rho L}{x_P W}$, где ρ – среднее удельное сопротивление; L – эффективная длина резистора; S – площадь поперечного сечения, равная ширине резистора W, умноженной на глубину перехода x_p для диффузионных и на толщину материала для поликремниевых и металлических.

В диффузионных резисторах из-за неоднородного распределения примесей трудно аналитическим способом определить среднее удельное сопротивление. В литературе приводятся подробные таблицы значений ρ для различных профилей распределения примесей и концентраций.

При практических расчетах резистора наиболее подходящим параметром является сопротивление слоя $\rho_{СЛ}$, которое можно определить как $\rho_{СЛ} = \frac{\rho}{x_P}$.

Сопротивление слоя выражается в омах и равно сопротивлению единичного квадрата данного материала. и выражается в Ом/□. Для данной величины $\rho_{СЛ}$ общее значение *R* можно выразить следующим образом: $R = \frac{\rho_{СЛ} L}{W}$.

Точность воспроизведения номинала резистора определяется точностью удельного сопротивления, точностью задания длины и ширины сопротивления. Обычно у резисторов $L >> W$, и неточность воспроизведения ширины сильнее влияет на разброс номиналов резисторов. Поэтому для повышения согласованности сопротивлений необходимо использовать достаточно широкие резисторы.

Резисторы с большим сопротивлением выполняют в виде секций, что позволяет уменьшить занимаемую площадь. Во многих случаях, например, в точных делителях напряжения, требуется большое количество резисторов, согласованных по величине. Для этого их лучше разбить на одинаковые элементарные резисторы, соединенные между собой металлическими шинами (рис. 5.3б).

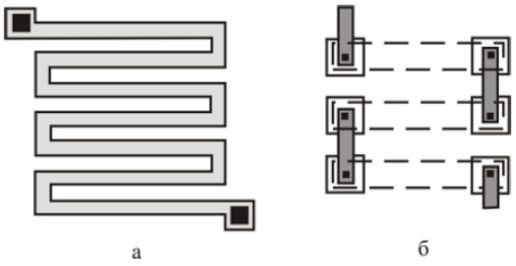


Рис. 5.3. Топология секционных резисторов: а – змейка из одного материала, б – сегменты резистора соединены между собой металлическими шинами

5.2.2. Конденсаторы

Интегральные конденсаторы сильно видоизменились по мере развития технологии ИМС. Самым доступным конденсатором в КМОП-технологиях является полевой транзистор, у которого исток, сток и подложка объединены и исполняют роль первой обкладки, а затвор является второй обкладкой (рис. 5.4а). Существенным недостатком является нелинейность транзисторной емкости из-за влияния области пространственного заряда (ОПЗ). Этого недостатка лишены поликремниевые конденсаторы (рис. 5.4б), в

которых обе обкладки выполнены из двух уровней поликремния, разделенных тонким слоем диэлектрика. Простейшая топология конденсатора представляет собой совокупность двух прямоугольников (обкладок), один из которых вложен в другой (рис. 5.5). Величина емкости конденсатора определяется формулой $C = \frac{\varepsilon_{OX} WL}{t_{OX}}$, где ε_{OX} – диэлектрическая проницаемость диэлектрика; t_{OX} – толщина слоя диэлектрика. Для снижения паразитной емкости в горизонтальных конденсаторах верхнюю обкладку конденсатора делают меньше нижней. Поликремниевые конденсаторы требуют наличия в технологии двух уровней поликремния, также они обладают большой паразитной емкостью на подложку нижней обкладки. В современных технологиях данный тип емкостей практически не используются.

Следующим этапом развития стало появление MIM-конденсаторов (*Metal-Insulator-Metal*) (рис. 5.4в). Конструктивной особенностью MIM-конденсаторов является наличие дополнительного уровня металла, электрически подключенного к верхнему металлу. Данный дополнительный уровень располагается вблизи предпоследнего металла, обеспечивая высокую удельную емкость. Паразитные емкости MIM на подложку значительно меньше, чем у поликремниевых, и чем больше уровней металла доступно в технологии, тем разница сильнее. Следующим преимуществом является возможность создания двойных и тройных MIM-конденсаторов, введением соответственно двух и трех дополнительных уровней металла, при этом существенно экономится площадь. В некоторых технологиях есть возможность увеличивать удельную емкость конденсаторов за счет изменения материала диэлектрика. Недостатком является необходимость введения дополнительных технологических операций, удорожающих производство ИМС. Однако на данный момент MIM-конденсаторы являются одним из основных типов емкостей, используемых при создании высокоточных схем и согласованных между собой массивов емкостей. Одной из проблем, связанной с MIM-конденсаторами, на заре их появления была частая закоротка обкладок.

Совершенствование фотолитографических процессов и способов плазмохимического травления позволили получать узкие полоски металла с малым расстоянием между ними. Так появились MOM-конденсаторы (*Metal-Oxide-Metal*), представляющие собой встречно-штыревую металлическую структуру (рис. 5.4г). В отличие от остальных типов конденсаторов, в которых обкладки расположены горизонтально, в MOM-конденсаторах используется преимущественно боковая емкость, а обкладки расположе-

ны вертикально. МОМ конденсаторы могут быть выполнены из нескольких уровней металла. Для дополнительного увеличения емкости и улучшения воспроизводимости используются МОМ-конденсаторы, где встречно штыревые структуры в соседних уровнях металлов ориентированы под углом 90° друг к другу. При необходимости использовать конденсаторы с заданным отношением емкостей их целесообразно набирать из идентичных элементарных конденсаторов, поскольку пропорциональное увеличение площади из-за краевых эффектов не обеспечивает пропорциональное изменение емкости (рис. 5.5).

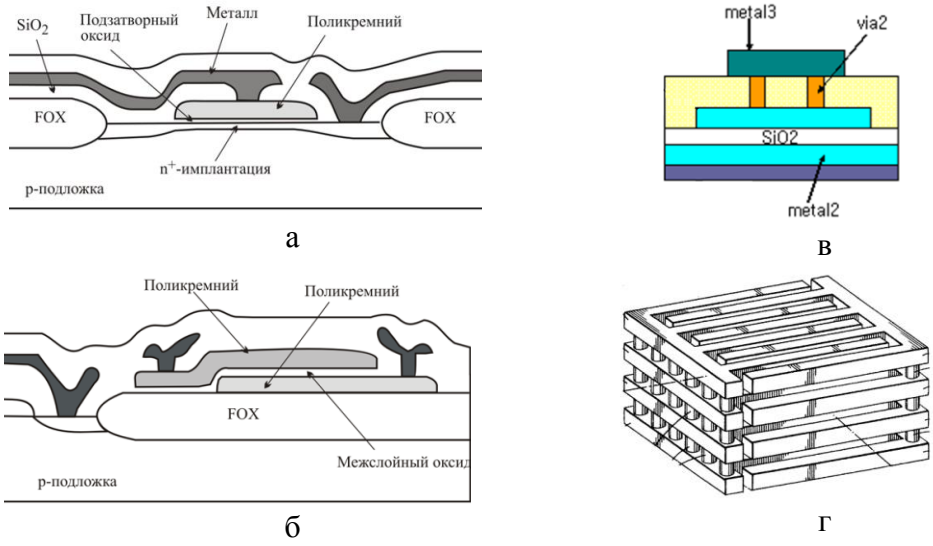


Рис. 5.4. Конденсаторы: а – МОП; б – поликремниевый; в – MIM; г – МОМ

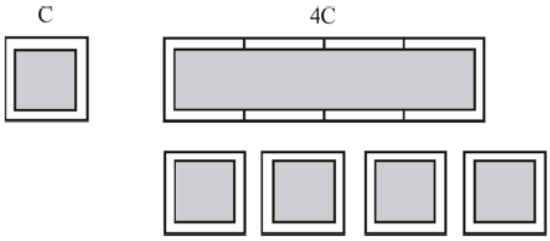


Рис. 5.5. Пропорциональное увеличение емкости конденсатора

5.3. Активные элементы

5.3.1. МОП-транзисторы

Полевой транзистор со структурой металл-окисел-полупроводник (МОП-транзистор) является основным элементом современных цифровых микросхем. Его

главным преимуществом с точки зрения цифровой обработки является прекрасная работа в качестве ключа и малое количество паразитных эффектов. Другие его важные преимущества состоят в достижимости высокой плотности интеграции при разумных затратах на производство. Различают два типа полевых МОП-транзисторов. *n*-МОП-транзистор содержит n^+-области истока и стока расположенные в подложке *p*-типа (рис. 5.6). На рис. 5.7 введены условные обозначения слоев: POLY1 – слой поликремния; *N*+ и *P*+ - области *N*- и *P*-диффузии соответственно; PSUB – подложка *p*-типа.

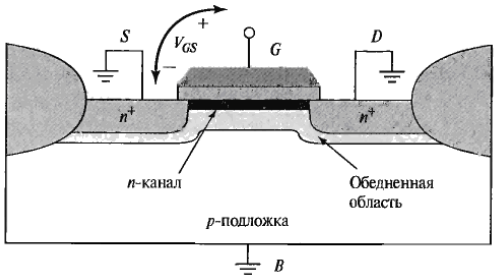


Рис. 5.6. МОП-транзистор n-типа с обедненной областью и индуцированным каналом при положительном напряжении V_{GS}

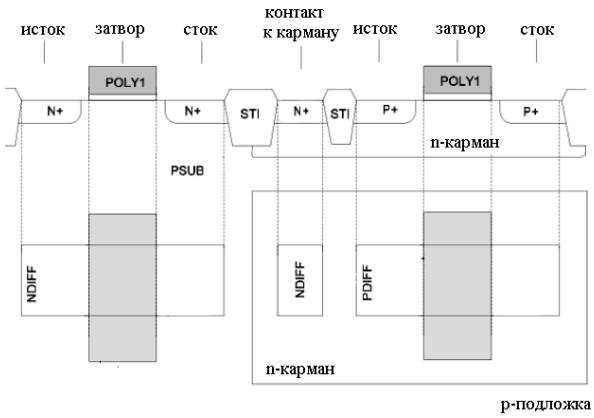


Рис. 5.7. Поперечное сечение структуры, формируемой в КМОП-процессе

Транзисторы с большой шириной могут быть разделены на маленькие и объединены последовательно по истокам и стокам. Разновидностями многозатворных МОП-транзисторов являются свернутый и гребенчатый МОП-транзисторы (рис. 5.8). Обычно объединение двух транзисторов выполняют по стоку, т.к. его паразитная емкость обычно выше истоковой. Поэтому свернутый транзистор при той же ширине канала имеет вдвое меньшую паразитную емкость стока и вдвое меньшее сопротивление затвора, что

особенно важно при создании высокочастотных схем. Гребенчатый МОП-транзистор (рис. 5.8б) используют при больших значениях ширины.

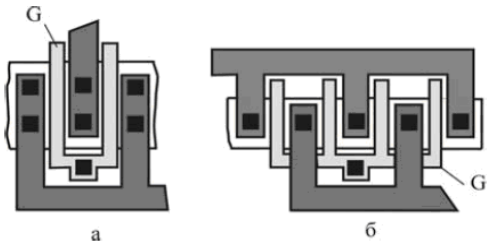


Рис. 5.8. Топология многозатворных МОП-транзисторов:
а – свернутого; б – гребенчатого

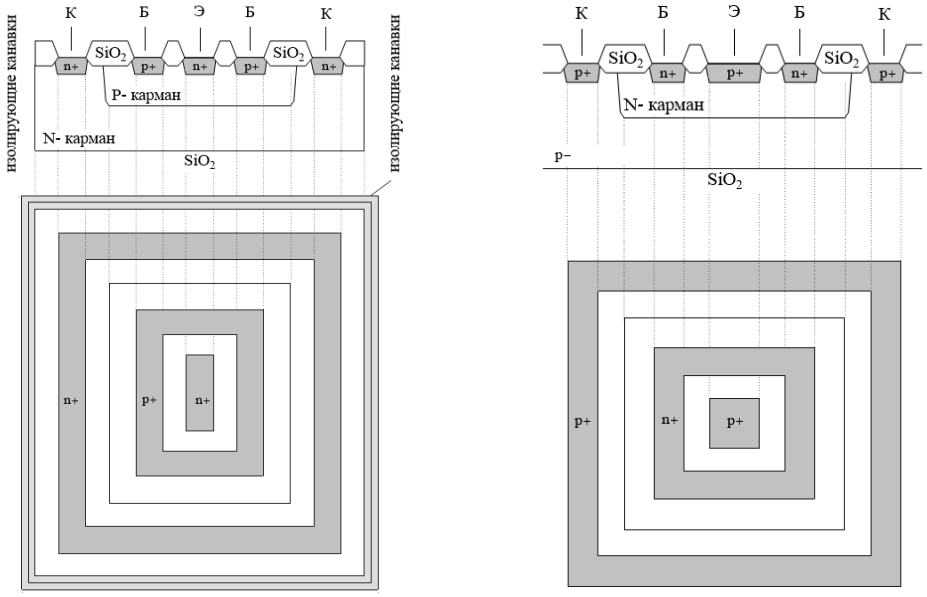
В аналоговых схемах к МОП-транзисторам предъявляются гораздо более строгие требования, чем в цифровых схемах. Это обусловлено и характером сигналов, принимающих непрерывный ряд значений, и необходимостью достижения высоких коэффициентов усиления. Первое условие накладывает ограничения на разброс характеристик транзисторов, а второе – на величину выходного сопротивления. Для удовлетворения таких требований в аналоговых ИС приходится использовать транзисторы с длиной канала в 1.5–2 раза, а в критических случаях в 4 раза превышающей минимальный топологический размер. Это неизбежно увеличивает размеры схем, ухудшает быстродействие и повышает потребляемую мощность.

Последовательное сопротивление стока и истока вызывает ухудшение быстродействия прибора, поскольку уменьшает ток стока при заданном управляющем напряжении. Таким образом, поддержание его значения на возможно низшем уровне является важной целью при проектировании, как для разработчика приборов, так и для разработчика микросхем. Во многих современных технологических процессах проводится силицидизация областей стока, истока и затвора, что позволяет существенно снизить величину поверхностного сопротивления. Другим способом снижения сопротивления является изготовление транзистора с большей шириной.

5.3.2. Биполярные транзисторы

Биполярный транзистор типа *n–p–n* является основным схемным элементом биполярных полупроводниковых интегральных микросхем. Он обладает лучшими характеристиками, чем транзистор типа *p–n–p*, а технология его изготовления более проста. Остальные элементы интегральной схемы выбираются и конструируются таким обра-

зом, чтобы они совмещались со структурой транзистора типа *n–p–n*. Однако в КМОП-технологиях обратная ситуация: приходится реализовывать биполярные транзисторы из диффузионных областей с той же концентрацией, что и полевые транзисторы, поэтому в КМОП-процессах либо полностью отсутствуют биполярные транзисторы, либо их параметры оставляют желать лучшего, например, коэффициент усиления β ~ (3–6). Примеры топологии и сечений *n–p–n*- и *p–n–p*-транзисторов представлены на рис. 5.9.



а б

Рис. 5.9. Топология и сечения биполярных транзисторов:
а - n–p–n-типа; б – p–n–p-типа

5.3.3. Диоды

В библиотеках элементов содержатся разнообразные диоды: от обычных диффузионных на переходах n+–подложка, p+–n-карман, n-карман–подложка, до диодов Шоттки и Зенера.

5.4. Вопросы для самопроверки

1. Какие виды пассивных элементов ИМС вам известны. Представьте их топологическое представление и сечение.
2. Какие виды активных элементов ИМС вам известны. Представьте их топологическое представление и сечение.

6. ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ КМОП-ИМС

6.1. Введение

Сложность разработки современных изделий микроэлектроники заключается в особенностях перехода проектных норм в субмикронный и нанометровый диапазоны, т.е. в существенном уменьшении топологических норм, резком увеличении общего количества цифровых ячеек. Расчет временных параметров, временная оптимизация, анализ цепей питания таких схем невозможен без применения современных высокоуровневых быстродействующих средств САПР.

Из этой главы вы узнаете:

- ✓ *Маршрут проектирования цифровых ИМС*
- ✓ *Зачем проверять логическое описание схемы*
- ✓ *Что происходит при компилировании нетлиста*
- ✓ *Зачем нужно учитывать силу драйвера*
- ✓ *Как осуществляется синтез дерева синхронизации*
- ✓ *Этапы работ программ place & rout*
- ✓ *Особенности трассировки критичных сетей*

6.2. Общий маршрут проектирования

В настоящий момент существует две основных методологии проектирования цифровых схем: методология логического синтеза и методология схемного описания. До недавнего времени методология схемного описания использовалась в большинстве разработок. Она представляет собой способ проектирования снизу вверх, когда проектируемая схема описывается на языке высокого уровня, начиная от нижних иерархических блоков и заканчивая верхними. При таком способе проектирования довольно трудно обеспечить требуемые характеристики изделий. Поэтому на смену этому способу пришла методология логического синтеза.

Общий маршрут проектирования можно представить как последовательность следующих этапов:

- ➢ системное проектирование – определяется общая архитектура изделия: набор блоков (ядро, память, периферийные модули); способы организации обмена дан-

ными; список выполняемых функций; внешний интерфейс; временные параметры и быстродействие; строится общая поведенческая модель;

- функциональное проектирование – осуществляется разработка высокоуровневой модели, представляющей собой RTL-код; функциональная верификация;
- физическое проектирование – синтез схемы и топологии устройства на основе разработанного RTL-кода; физическая верификация и моделирование.

6.3. Проверка логической схемы.

Обычно разработчики для создания огромных цифровых схем используют VHDL или Verilog описания. VHDL (*Very High Speed Integrated Circuits Hardware Description Language*) – стандарт IEEE с 1987 года, похож на язык Pascal и разработан по заказу Министерства обороны США. Язык Verilog разработан в 1985 году фирмой Gateway Design Automaton, позднее поглощенной Cadence. Verilog базируется на языке C, не имеет возможности создания новых типов данных, но более прост в реализации и лаконичен, что позволяет уменьшить объем описания схем по сравнению с VHDL. На данных языках создается файл, описывающий функцию или поведенческое описание разрабатываемой схемы.

Язык Verilog поддерживает три основных уровня абстракции: поведенческий, RTL-уровень (*Register Transfer Level* – уровень межрегистровых передач) и структурный. Поведенческий уровень описывает функционирование системы в целом, без учета внутренней архитектуры. На RTL-уровне поведение схемы определяется в терминах потока данных между регистрами и логических операций над данными (комбинаторная логика). На структурном уровне проект представляется в виде иерархии компонентов. Различные уровни абстракции представлены на рисунке 6.1.

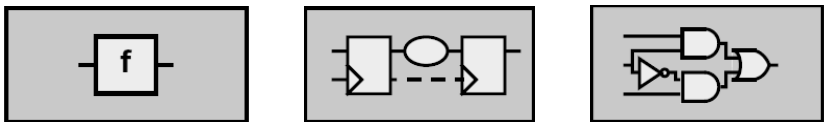


Рис. 6.1. Уровни абстракции: а – поведенческий; б – RTL-уровень; в – структурный

Основной структурной единицей проекта на языке Verilog является модуль. Он описывается ключевыми словами *module – endmodule*, что соответствует *entity – end* в VHDL. Как правило, модуль содержит список портов – интерфейсных сигналов, кото-

рые служат для подключения его к другим модулям. Порты бывают трех типов: *input* – входы, *output* – выходы, *inout* – двунаправленные. Ниже приведен общий синтаксис модуля.

module имя_модуля [список портов];
[output // выходы
inout // двунаправленные порты
input] // входы
// Тело модуля
initial
begin
..
end
always
..
endmodule

Символами // обозначается однострочный комментарий. Verilog также поддерживает многострочный комментарий /* */.

В модуле используются процедурные блоки *initial* и *always*. *Initial* служит для описания действий, которые выполняются один раз, например, при запуске модели, а always обозначает действия, которые выполняются многократно, на протяжении всего времени. Ключевые слова *begin – end* служат для объединения последовательно выполняемых операторов в единый с точки зрения синтаксиса оператор. В противоположность последовательному блоку *begin – end* блок *fork – join* объединяет параллельные операторы. Значения задержек в нем отсчитываются не относительно предыдущего оператора, а относительно начала блока.

Для моделирования аппаратных средств Verilog поддерживает 4-х значную логику: 0, 1, z, x. Первые два значения соответствуют логическим уровням, третье – состоянию с высоким импедансом, четвертое означает неопределенное состояние и используется при моделировании неинициализированных сигналов, конфликтов, метастабильных состояний триггеров – во всех случаях, когда симулятор не может определить значение данного сигнала.

Все типы данных в языке Verilog можно разделить на две основных группы: переменные и цепи. Цепь представляет собой физическое соединение цифровых ячеек. К

цепям относятся *trireg, wire, wand, wor, supply1, supply0*. В отличие от цепей переменные сохраняют значение до следующего присваивания.

Verilog поддерживает следующие типы переменных:

reg – беззнаковая переменная (самый распространенный тип);

integer – 32-разрядная переменная со знаком;

real, realtime – вещественные переменные двойной точности;

time – 64-разрядная беззнаковая переменная, которая используется для сохранения времени моделирования.

Так как Verilog используется для моделирования физических систем, то большое внимание уделено привязке события к определенному моменту времени. Для этого используется механизм задержек. Конструкция # позволяет приостановить исполнение оператора на заданное время. Например, выражение #10 *a = b* присвоит переменной *a* значение *b* спустя 10 временных единиц. Вместо числа могут использоваться параметры, например при задании тактового сигнала:

```
parameter real period = 10.0;
..
always @(posedge clk)
  #(period / 2) clk = ~ clk;
```

Конструкция @(список чувствительности) представляет собой событийный контроль, который задерживает исполнение последующих операторов до тех пор, пока не произойдет событие из списка чувствительности. В качестве событий могут использоваться изменения сигналов – цепей и переменных, положительный (*posedge*) или отрицательный (*negedge*) фронт сигнала.

Оператор *wait*(*expression*) позволяет задержать исполнение последующих операторов, пока выражение expression не станет истинным. Если выражение уже истинно, то выполняется следующий оператор.

Для описания устройства на высоком уровне абстракции Verilog предоставляет следующие поведенческие конструкции: условный оператор *if – else*, оператор выбора *case* и его разновидности, циклы *for, forever, repeat, while*.

Операторы *if–else* и *for* работают так же, как и в других языках программирования. Оператор *case* (*expression*) позволяет сделать выбор из нескольких вариантов:

```
case (sel)
  3: y = d;
```

2: y = c;

1: y = b;

default: y = a;

endcase // sel

Если ни одно из условий не совпадает, выполняется ветвь *default*.

Цикл forever выполняется на протяжении всего времени моделирования, если не задан список чувствительности:

forever @(interrupt_event); // реагирует на каждое событие interrupt_event.

Цикл repeat исполняет следующий оператор заданное число раз:

repeat (10) @(posedge clk); // ожидаем в течение 10 периодов сигнала clk

Цикл *while* (*expression*) выполняет последующий оператор до тех пор, пока выражение *expression* истинно. Если выражение ложно изначально, то цикл не выполняется ни разу:

while (a > 0)

a <= a – 1;.

После написания RTL-кода его необходимо проверить. Для верификации модели создается тестовое окружение (*testbench*) – это модуль верхнего уровня, в котором на испытываемое устройство DUT (*device under test)* или DUV (*device under verification*) подаются входные воздействия и проверяется реакция на эти воздействия на выходах. Обобщенная модель тестового окружения представлена на рис. 6.2. Тестирование происходит намного быстрее, чем на транзисторном уровне. Заслуженной популярностью пользуются симуляторы фирмы Cadence Verilog-XL и NC-Verilog.

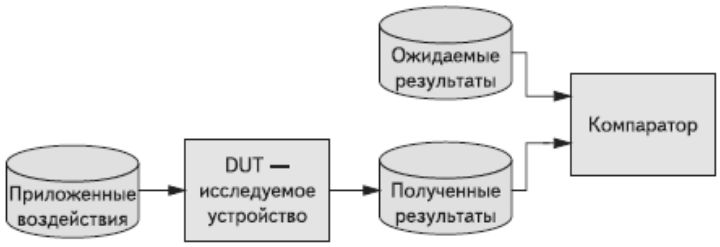


Рис. 6.2. Обобщенная модель тестового окружения

При моделировании проекта симулятор выполняет следующие шаги:

- компиляция - симулятор считывает Verilog-описание, обрабатывает директивы и строит иерархию проекта;

- инициализация – на этом этапе инициализируются параметры модуля, соответствующие переменные (*reg, time* и *integer*) устанавливаются в неизвестное состояние, а неуправляемые цепи – в Z-состояние;

- симуляция – симулятор обрабатывает все события, расписанные на текущий момент времени моделирования, затем увеличивает время (второй параметр директивы *`timescale*) и продолжает вычисления. Моделирование останавливается, когда в очереди больше нет событий.

Симуляторы нуждаются в технологически зависимом описании каждой логической функции. Эта информация помещается в серии файлов. Кроме электрического описания есть логическое представление каждого логического элемента, который также использует симулятор. Все эти файлы называются стандартной библиотекой ячеек (*standard cell library*) или библиотекой логических элементов (*logic library*). При моделировании VHDL можно внести необходимые коррективы до проработки топологии, что позволяет значительно экономить время разработки.

6.4. Компилирование нетлиста

После тестирования кода разработчик компилирует код в *silicon compiler* или *logic synthesizer*, которые транслируют код высокого уровня в файл, содержащий необходимые логические функции и информацию об их соединениях между собой. Для синтеза электрической схемы используется библиотека элементов конкретного технологического процесса. Самыми известными продуктами являются RTL Compiler фирмы Cadence и Design Compiler (Synopsys).

При компиляции кода разработчик контролирует следующие параметры: площадь, мощность и скорость. В зависимости от требований разработчик отдает приоритет тем или иным параметрам. Может потребоваться введение дополнительных временных ограничений, например, для цепей синхронизации. После синтеза схемы RTL Compiler проводит оптимизацию схемы с целью устранения критичных путей, а также оптимизацию дерева синхронизации. Поэтому конечный результат зависит от приоритетов. Этап проверки функциональных параметров и характеристик схемы на логическом уровне называется *prelayout simulation*. Верификацию нетлиста проводят в том же тестовом окружении.

6.5. Сила драйвера

Компилятор может создать очень большие сети, к которым подключены несколько сотен или тысяч транзисторов. Чем больше транзисторов в сети, тем большая мощность нужна для их переключения. Таким образом, прежде чем перейти к процессу синтеза топологии, нужно модифицировать нетлист, чтобы быть уверенным, что большие сети прокачиваются соответствующим источником (драйвером). То есть, необходимо заменить имеющийся драйвер на такой же по логической функции, но более мощный (*drive strength* или *fan out*) (рис. 6.3). Fan out – это число затворов, которые может прокачать драйвер. Например, в библиотеке содержится большое число однотипных элементов маркированных 1x, 2x или 4x, что означает, что они обеспечивают переключение 2-х, 4-х или 8-и затворов.

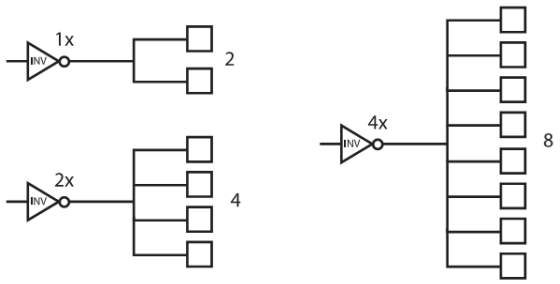


Рис. 6.3. Нагрузочная способность инверторов 1x, 2x и 4x

Чтобы не потреблять лишнюю энергию и площадь кристалла, нужно знать на какую нагрузку работает драйвер. При анализе схемы компилятор будет дробить большие сети на маленькие, которые сможет прокачать драйвер. При дроблении потребуются ячейки, не имеющие логической функции. Данные ячейки, называемые буферными, расставляются автоматически. При очень большом числе дополнительных буферов, они станут вносить значительные задержки, которые не учитывались при начальном моделировании. Поэтому после расстановки буферов необходимо заново промоделировать компилированный нетлист.

6.6. Синтез дерева синхронизации

Принцип работы цифровых устройств построен на тактировании специальным сигналом, называемом тактовым, клоковым (*clock*) или синхросигналом. Переключение всех сигналов осуществляется по фронтам тактового сигнала. Современные устройства

содержат огромное количество элементов, измеряемое миллионами. Вся масса элементов должна переключаться синхронно. Именно здесь и проявляется проблема тактирования из-за возникновения паразитных эффектов, мешающих нормальной работе схемы. Как правило, источник синхронного сигнала находится далеко. Шина, передающая сигнал *clock*, называется клоковой сетью или сетью синхронизации, которая, как правило, очень большая. Существуют различные способы построения деревьев синхронизации, различающиеся принципами построения и функционирования. Независимо от способа реализации дерева синхронизации тактовый сигнал должен оставаться стабильным во всех частях схемы, фронт тактового сигнала должен приходить ко всем элементам схемы одновременно.

Для сигналов синхронизации важно не только их одновременное воздействие на регистры состояний, но и длительности фронтов, которые не должны быть больше критической величины для выбранного типа триггеров. Трудно сделать источник, способный прокачать все затворы. Поэтому сеть синхронизации нужно перевести в функциональную, разделив ее на малые подсети и добавив буферные ячейки. Таким образом, сеть преобразуется на ветвистые участки, которые называются деревом синхронизации (*clock tree*). Например, есть сеть из 6 ячеек, а библиотека содержит элементы, которые способны прокачать максимум 3 ячейки. В таком случае разбиение исходной сети (рис. 6.4а) будет выглядеть как на рисунке 6.4б. Если же сеть значительно больше, компилятор продолжит дробление сети и добавление буферов, к каждому из которых подключено не больше 3 элементов (рис. 6.4в).

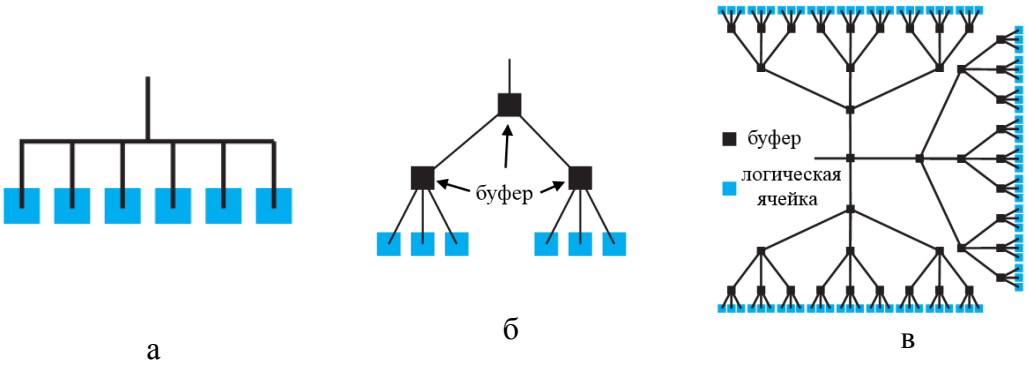


Рис. 6.4. Разбиение сети синхронизации: а – исходная сеть без буферов; б – разбиение исходной сети на подсети; в – разбиение более крупной сети на маленькие в соотвествии с нагрузочной способностью буфферов

Один из алгоритмов построения дерева синхронизации в общем виде выглядит так: площадь кристалла или цифрового блока, занятая логическими элементами, разбивается на квадранты. Каждый квадрант делится на более мелкие квадранты и т.д. В первую очередь тактовый сигнал подается в центр кристалла. Корневой драйвер управляет четырьмя драйверами второго яруса, размещенными в больших квадрантах. Далее сигнал идет к драйверам третьего яруса, размещенным в малых квадрантах и т.д. Необходимое число ярусов определяется сложностью схемы и нагрузочной способностью драйверов. Длины проводников в каждом ярусе по возможности выравниваются. Синхросигнал от источника доходит до каждого триггера через одинаковые элементы и линии связи, что обеспечивает минимальный разброс задержек фронтов импульсов. В идеальном случае, если путь отлично сбалансирован, расфазировка тактовых импульсов равна нулю. Хотя на распространение сигнала от центральной точки к каждому конечному узлу может требоваться несколько тактов, на все узлы тактовые сигналы поступают одновременно. В реальности из-за разбросов технологических процессов и влияния паразитных эффектов в схеме возникают расфазировка и дрожание сигнала. Основной метод обеспечения синхронизации сигналов при физическом проектировании – это анализ системы с учетом размещения элементов и реальных параметров проводников.

Распространение тактового сигнала необходимо рассмотреть на начальных стадиях разработки сложной схемы, поскольку это может существенно повлиять на топологию кристалла. Если рассматривать дерево синхронизации только в конце цикла разработки, когда основная часть топологии кристалла зафиксирована, то неоптимальные цепи распространения тактовых импульсов могут снизить быстродействие конечной схемы. Когда процесс отладки нетлиста завершен, все будет готово для синтеза топологии.

6.7. Синтез топологии

Целью топологического проектирования является получение рабочей топологии устройства, обеспечивающей функциональные характеристики в диапазоне заданных условий (частота, температура и др.) и электрические параметры в соответствии с техническим заданием (напряжения питания, токи, нагрузочная способность и др.). Одной из самых распространенных платформ для синтеза топологии является Encounter. Для получения качественных технических и функциональных характеристик требуется

строгое соблюдение правил проектирования топологии. Разработка топологии происходит в несколько этапов.

1. Этап планирования кристалла (*Floorplanning*). На данном этапе оценивается размер будущего кристалла, расположение контактных площадок, размещение блоков по кристаллу, предварительная плотность элементов.

2. Этап размещения блоков и базовых библиотечных элементов (Placement) и подключение питания (*Special Routing*). На этом этапе проводится размещение периферийных колец питания и основных блоков, построение колец питания вокруг блоков, располагаются горизонтальные и вертикальные шины питания (*stripes, rails*).

3. Этап трассировки межсоединений (*Routing*). На данном этапе выполняется построение дерева синхронизации, осуществляется трассировка всех межсоединений в соответствии с нетлистом.

4. Этап временного анализа и оптимизации. Синтезатор оптимизирует структуру дерева синхронизации с целью улучшения временных параметров работы схемы, подгоняя ее под заданную частоту.

5. Этап энергетической оптимизации. Этап предназначен для всестороннего анализа цепей питания: оценка общего, поблочного и критического энергопотребления, статический и динамический анализ сетки питания.

6. Этап физической верификации включает проверку соответствия топологии правилам проектирования и нетлисту.

7. Построение полного списка соединений (нетлиста) и извлечение файла задержек с целью осуществления всесторонней проверки работоспособности топологии устройства.

Рассмотрим основные этапы синтеза топологии подробнее.

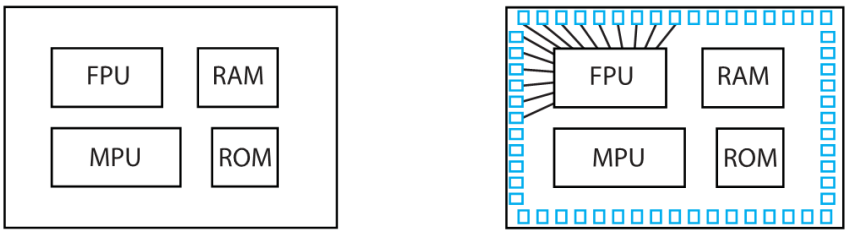
6.7.1. Floorplaning

Первой подпрограммой является floorplanning tool. Она позволяет выделить функциональные области, подключения между ними, расположение IO-ячеек и предложить вариант с наиболее простой трассировкой.

Обычно блок делится на функциональные области. Например, микропроцессорная часть кристалла MPU, блоки с плавающей точкой FPU, блоки памяти RAM, ROM, и т.д. Далее каждую функциональную область располагают по своему усмотрению. Например, микропроцессорную часть размещают в нижнем левом углу, RAM – в пра-

вом верхнем и т.д. (рис. 6.5а) Перестановки можно проводить многократно и видеть, как они влияют на межсоединения.

После определения функциональных областей необходимо сгруппировать цифровые ячейки каждого из блоков в какой-либо области (рис. 6.5б). Например, чтобы управляющие сигналы одного блока легче было подключить к другим блокам или IO-ячейкам. На данном этапе не важна детализация, достаточно сгруппировать ячейки.

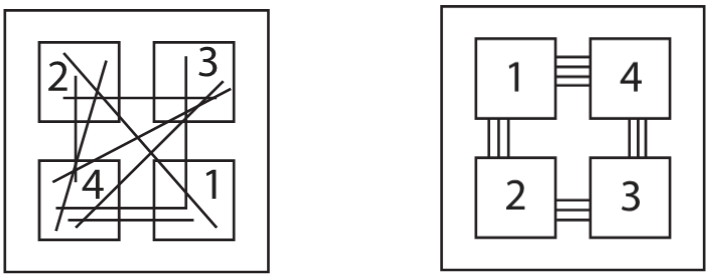


а б

Рис. 6.5. Пример Floorplan кристалла: а – выделение функциональных блоков; б – опимизация расположения входов и выходов блоков

При перемещении блоков будут отображаться межсоединения между ними. Если есть множество пересечений и соединения беспорядочны, то реализовать такой вариант топологии будет достаточно сложно (рис. 6.6а). Необходимо добиваться, чтобы floorplan выглядел максимально просто (рис. 6.6б). Для этого допустимо переразбивать функциональные области, переносить маленькие блоки.

Когда файл с floorplan сохранен, и известно где будут размещены цифровые ячейки, placement tool определяет длины сетей, которые далее учитываются при моделировании. Если влияние существенно, следует модифицировать floorplan и нетлист, например, добавив более мощные драйверы. Когда задержки по шинам перестают оказывать влияние на работу схемы, наступает этап тонкой настройки топологии.



а б

Рис. 6.6. Метод графов для определения качества floorplan: а – плохой вариант; б – хороший вариант

6.7.2. Размещение (placement)

Placement tool фиксирует положения элементов. При выделении блока будут видны ячейки, и можно выполнить их перегруппировку. На этапе placement расставляются транзисторы не только блоков, но и IO-драйверов. Для этого используется отдельный tool, который дополнительно использует ESD-правила. После размещения блоков и ячеек можно приступить к трассировке.

6.7.3. Трассировка (routing)

Трассировка имеет две приоритетные сети – питание и тактовый сигнал. Поэтому подключение начинается с них. Далее следуют подключения в соответствии со схемой, начиная с критичных сетей, которые указываются заранее.

6.7.3.1. Трассировка шин питания

Цепи питания – важнейшая структурная часть топологии кристалла, от конфигурации и реализации которой в большой степени зависят электрические и функциональные параметры схемы, и даже само функционирование устройства. Цепи питания разделяются на глобальные (питание ядра с блоками) и питание периферии (IO-ячеек), называемое обвязкой. Глобальные цепи питания определяются из оценок потребления тока блоками, что влияет на ширину колец цепей питания и число контактных площадок. Обвязка – важная часть области кристалла. Периферийные кольца подключаются к контактным площадкам питания. Сначала размещаются периферийные ячейки, затем крупные блоки. Вокруг каждого блока в свою очередь формируют свои кольца питания и земли. Шины должны быть сконцентрированы в определенных местах и направлении. Можно корректировать шины, например, расширяя их в областях повышенного потребления.

На рисунке 6.7а представлена ситуация, когда верхняя правая ячейка находится дальше от пада питания в сравнении с левой нижней. Сопротивление до первой ячейки будет значительно меньше, если добавить параллельные шины – *strap*-ы, снижающие сопротивление (рис. 6.7б). В настройках программ задается шаг стрепирования, например, через 60 или 100 мкм.

Трассировка цепей земли и питания может осуществляться верхним толстым или нижними слоями металлов. Подключение в верхнем слое выполняется, как правило, при проектировании схем с высокой плотностью элементов.

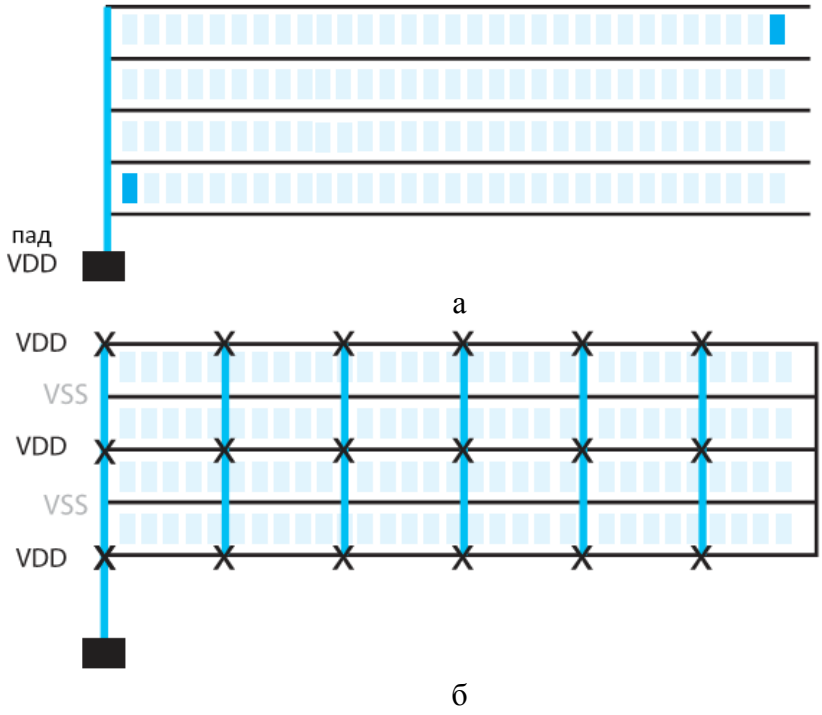


Рис. 6.7. Подключение питания к двум ячейкам:
а – до страппирования; б – после страппирования

При проектировании цепей питания и земли в нижних слоях (обычно применяется в технологиях с небольшим числом металлов 3-4) непосредственно после расположения IO-ячеек осуществляется планирование периферийных колец питания. Сразу после этого располагают макроблоки и формируют вокруг них кольца земли и питания, подключая их к периферийным питающим кольцам. Далее размещаются базовые ячейки и достраиваются все горизонтальные и вертикальные линии питания (рейлы и страпы).

После проектирования топологии устройства проводится анализ сетки питания. Осуществление power-анализа – обязательное условие подтверждения работоспособности ИМС. Различают два типа анализа: статический и динамический.

Статический анализ показывает места топологии, которые наиболее подвержены влиянию падения напряжения, а также служит для оценки качества сетки питания и определения запаса по мощности.

Эффект падения напряжения на шинах питания может возникать из-за повышенного сопротивления шины, а также в результате того, что шина рассчитана на протекание меньшего тока чем требуется. К повышению сопротивления может привести большая длина шины, большое количество межслойных переходов. Для анализа сетки питания может быть использовано приложение Power System платформы Encounter САПР

Cadence. Анализатор разбивает топологию на участки, протягивающиеся от входа питания до потребителей, которые затем анализируются на наличие просадки напряжения питания при известных нормах потребления элементов и вычисленных нагрузках. В результате анализа строится картинка, представляющая собой топологию с окрашенными в различный цвет областями. Каждый цвет соответствует определенному диапазону падений напряжения на шинах питания. Как правило, максимальное падение напряжения питания не должно превышать 5-10% от номинального. В случае обнаружения отклонений от нормы просадки требуется оптимизировать топологию или провести синтез топологии заново. Статический анализ может выявить таким образом «узкие» места сетки питания. Приложение Power System предлагает два способа оптимизации.

Первый способ оптимизации основан на анализе возможностей улучшения сетки питания за счет улучшения контактирования слоев. Приложение анализирует сетку питания и находит места, где одноименные шины питания или земли расположены в различных слоях друг над другом и предлагает добавить межсоединения. При этом улучшается общее контактирование нагруженной шины, уменьшается электрическое сопротивление, а, следовательно, и падение напряжения в шине питания. Также этот способ оптимизации включает анализ контактирования подложки.

Второй способ оптимизации предполагает расстановку так называемых развязывающих буферных конденсаторов (*decoupling capacitor*), размещаемых под шинами питания в свободных местах и подключаемых к сетям питания. Просадка напряжения уменьшается за счет использования накопленного в конденсаторах заряда в момент роста тока потребления. Чем больше общая емкость таких буферных конденсаторов, тем меньше возможность появления критичной просадки. Однако при наличии достаточно большого количества буферных конденсаторов может возникнуть резкий скачок тока потребления при подаче питания, связанный с одновременным началом заряда всей массы емкостей. Для устранения этого эффекта используется схема распределенного во времени включения блоков и схема ограничения подачи питания на различные участки сетки питания.

Динамический анализ позволяет протестировать сетку питания схемы в рабочем режиме. Существует два возможных направления анализа.

В первом случае система анализа проверяет нагрузку на шины питания и просадку напряжения при условии переключения определенного процента (обычно около 20%) блоков и ячеек, подключенных к этим шинам. Данные о потребляемой мощности

для каждой ячейки берутся из технологических файлов библиотеки. Анализ проводится непосредственно в момент переключения. Частота переключений зависит от тактовой частоты схемы. В результате выполнения анализа на экран выводятся данные с областями топологии с различным уровнем потребления и падения напряжения, зависящие от сопротивления шин и передаваемой по ним мощности.

Второй способ требует наличия тестов, проверяющих работу схемы. Тестовое окружение должно быть по возможности более полным, учитывающим все нюансы работы схемы, критичные пути. Полнота тестового покрытия при этом может подтверждаться соответствующими программами анализа. Динамический анализ выполняется во время работы в зависимости от заданных воздействий. При таком способе не требуется ограничение количества одновременно переключаемых ячеек и блоков, т.к. схема функционирует в рабочем режиме, близком к реальному.

6.7.3.2. Трассировка дерева синхронизации

Следующим этапом трассировки является создание дерева синхронизации. Основная цель – достижение как можно более синхронного прихода фронта тактового сигнала на каждый из элементов схемы. Трассировщик просчитывает все задержки на пути распространения тактового сигнала и оптимизирует этот путь, расставляя элементы задержки. Окончательная оптимизация дерева синхронизации выполняется после завершения полной трассировки топологии. Существует несколько приближений к реализации дерева синхронизации.

Приближение центральной магистрали. Обычно есть драйвер, сила которого достаточна, чтобы прокачать верхний уровень буферов. Этот драйвер располагают в центре и создают центральную шину, к которой подключают буферы верхнего уровня (рис. 6.8). Эта шина распадается на более мелкие ветки.

Анализ дерева синхронизации применяется с целью выявления уровней задержек и оптимизации топологии таким образом, чтобы задержки распространения фронтов тактового сигнала были минимальными. При анализе учитывается время прихода фронта синхросигнала в каждую точку схемы. Для успешного захвата данных триггером, данные должны поступить на вход триггера за определенное время до прихода фронта тактового сигнала, называемого временем установления данных, и продержаться дополнительно некоторое время после окончания фронта сигнала, называемое временем

удержания данных. Трассировщик анализирует и оптимизирует задержки тактового сигнала, чтобы удовлетворить этим требованиям. Формирование задержек осуществляется с помощью встраивания в цепь синхронизации буферных ячеек. Тактируемые цепи, расположенные рядом со схемой формирования тактового сигнала требуют введения больших задержек, относительно цепей расположенных на максимальном удалении от источника сигнала.

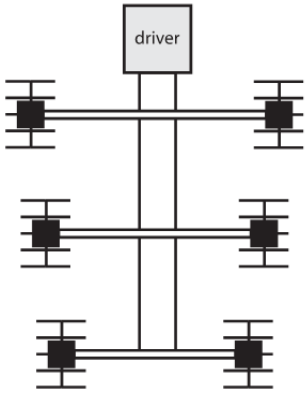


Рис. 6.8. Приближение центральной магистрали дерева синхронизации

Оптимизация проходит в несколько шагов, на каждом из которых оптимизатор изменяет конфигурацию дерева синхронизации, встраивает или убирает буферы задержек, улучшая общие показатели схемы по задержкам. При этом просчитываются все пути распространения тактового сигнала, и выявляется наиболее длинные и критичные. Программа оптимизации учитывает наихудшие условия, возникающие в результате внешних воздействий и технологических отклонений, такие как: максимальные задержки элементов схемы, максимальные паразитные значения элементов трассировки и зависящие от них значения задержек.

6.7.3.3. Трассировка прочих сигналов

Можно указывать критичные сети, подключать их вручную, либо задать к ним определенные требования, например кратчайшие пути трассировки для высокочастотных сигналов, полный или частичный запрет трассировки над определенными областями. Далее проводится трассировка прочих сетей.

Трассировка больших кристаллов может занимать несколько дней, после ее завершения может потребоваться ручная доводка. Например, удаление лишних петлей,

либо завершение сетей, которые программа не осилила. Иногда трассировка невозможна из-за слишком большого числа шин, поэтому приходится возвращаться к этапу floorplan.

После успешного завершения трассировки и оптимизации создается список задержек, в который содержатся задержки по всем элементам и цепям топологии. Список представляется в виде sdf-файла и содержит максимальные (наихудшие) значения задержек. После формирования списка осуществляется моделирование схемы, извлеченной из топологии в виде Verilog-нетлиста, с учетом выявленных задержек. При высоком уровне качества характеризации библиотеки точность результатов моделирования, как функциональных характеристик, так и электрических параметров схемы очень высока, что означает практически полное соответствии модели реальной схеме по перечисленным параметрам. Описание работы топологии с помощью файла задержек, определение временных и электрических зависимостей по входам схемы называется временной характеризацией.

6.8. Возможные проблемы трассировки и способы их устранения

В идеальном случае основной задачей приложения-трассировщика является конфигурирование и трассировка шин питания и земли, цепей тактирования и прочих сигнальных проводников по кратчайшему пути, с минимальным использованием переходных межслойных контактов. Однако при трассировке реальных схем с высоким коэффициентом утилизации и малым количеством металлов, используемых для трассировки, эти условия очень часто являются невыполнимыми. В этих случаях могут проявляться нежелательные физические явления, такие как, появление перекрестных искажений, разбег фронтов сигналов и другие.

В современных электронных схемах имеют место всевозможные эффекты, проявляющиеся при проектировании и приводящие к неправильному функционированию интегральных схем или искажению формы сигналов. Наиболее заметными из этих физических эффектов являются:

- шумы и задержки как следствие перекрестных искажений;
- падение напряжения на внутреннем активном сопротивлении;
- электромиграция;
- индуктивность.

Перекрестные искажения возникают из-за взаимной емкостной связи между проводниками схемы, в результате чего при изменении уровня сигнала в проводнике форма сигнала в соседних проводниках изменяется. Эти эффекты существенно проявились с переходом к субмикронным технологиям из-за значительного повышения емкостной связи между проводниками, за счет уменьшения расстояния между проводниками. Если в результате скачков напряжения в цепи-приемнике наводится сигнал, близкий к уровню переключения логического состояния, это может вызвать срабатывание последующих устройств, что в свою очередь приводит к неправильному функционированию схемы.

Проблемы подобного характера желательно идентифицировать до трассировки еще на этапе размещения блоков и элементов, например, указать максимально допустимое значение длины двух параллельных проводников на этапе трассировки. Но указанное ограничение способно существенно повлиять на возможность трассировки схем, усложнить ее. Частично обойти данную проблему можно путем создания ограничений на трассировку некоторых критичных шин (например, высокочастотных цепей), прокладкой критических цепей ручным способом с использованием кратчайших путей.

Одной из основных является проблема падения напряжения, в результате снижается быстродействие. При переходе к субмикронным технологиям эта проблема стала еще более актуальной из-за следующих факторов:

- увеличения токов, связанного с увеличением числа устройств в проекте;
- увеличения сопротивления проводников за счет уменьшения их сечения и числа межсоединений;
- существенное снижение напряжений питания.

Для уменьшения степени влияния или полного исключения проблем, связанных с питанием схемы, требуется соблюдение следующих рекомендаций при трассировке шин питания:

- создание широких периферийных колец;
- создание колец достаточной ширины вокруг блоков схемы;
- качественное подключение блоков к питанию, например, за счет подключения колец вокруг блоков к нескольким местам периферийных колец;
- частая сквозная прошивка вертикальными шинами питания всей площади кристалла;
- выделение для подключения питания верхнего толстого слоя металла;

- равномерное размещение блоков по всему кристаллу, исключение трассировки питания длинными шинами;
- использование приложений статического и динамического анализа после выполнения трассировки.

Увеличение плотности тока, протекающего через проводники, породило проблему электромиграции. Эффект электромиграции проявляет себя в местах повышения плотности тока в течение долгого времени. При этом происходит разрушение металла и появление обрывов и, как следствие, отказ всего устройства. Современные средства САПР позволяют оценить плотности токов, как в шинах питания, так и в сигнальных цепях на этапе предварительной трассировки.

Следующей, немаловажной проблемой, возникающей при переходе к современным субмикронным технологиям, является проблема возникновения индуктивных эффектов, способных причинить вред высокочастотным устройствам. Эффект самоиндукции является важным для длинных цепей, а эффект взаимоиндукции влияет на длинные параллельные участки шин. Чтобы снизить индуктивные эффекты, наиболее важно обеспечить правильный путь возвращения высокочастотных токов в общую шину. Эта задача чрезвычайно затруднена для автоматических инструментов размещения и трассировки, так как в настоящее время нет достаточно точного механизма анализа происходящих в них эффектов. Существуют специальные рекомендации по проектированию подобных схем. При трассировке шин необходимо дополнительно экранировать цепи дерева синхронизации. Подобным образом поступают с сетями высокочастотных синхросигналов в чувствительных блоках и частях схем, размещая шины тактовых сигналов между шинами земли и питания.

6.9. Верификация

Файл, содержащий данные о паразитных элементах отправляется обратно разработчику для моделирования. Теперь возможна верификация не с оценками, а с реальными значениями паразитных элементов. Если изначально floorplan был не удачным, то, скорее всего, последуют правки. Вплоть до этого момента работа с реальными транзисторами не проводилась. Все действия проводились только с прямоугольниками, где написано "здесь вход", "здесь выход". Если заменить абстрактные компоненты, на компоненты реальной библиотеки, добавить данные о положении и межсоединениях, то

получится выходной GDSII файл. После получения GDSII файла необходимо проверить корректность и полноту подключений. Для этого сначала проверяются технологические правила с помощью программного обеспечения DRC (*Design Rule Check*). Затем проверяется соответствие топологии нетлисту с помощью программ LVS (Layout Versus Schematic) (рис. 6.9). Проверка работоспособности схемы с учетом вычисленных задержек называется postlayout simulation. Моделирование может осуществляться на базе приложения NC Verilog.

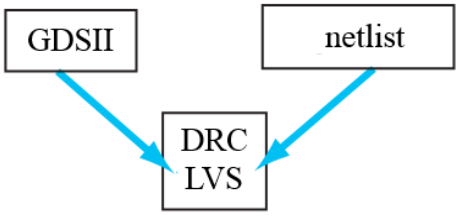


Рис. 6.9. Проверка GDSII файла

Что может пойти не так при высокой автоматизации процесса? Многое! Процесс расстановки и трассировки реализуется за счет синхронизации всех многочисленных баз данных для каждой подпрограммы. Существенным становится управление библиотеками. Например, в топологии инвертор может называться Inv1x, но LVS не запускается, т.к. в названии схематика сделана ошибка. Программы считают, что все базы актуальны, если были выполнены обновления в одном блоке, то может потребовать обновление в 6-7 ячейках выше по иерархии в различных местах, а если этого не сделано, то могут появиться места, где шины болтаются не подключенными. При наличии огромного числа элементов требующих 100% синхронизацию очень легко сделать ошибку.

6.10. Блок-схема проектирования цифровых ИМС

Сначала описывается логика работы, затем синтезируется нетлист, разрабатывается floorplan, затем несколько раз повторяются проверки логики работы. Затем расставляются элементы, и проводится их трассировка, выполняется проверка влияния задержек на функционирование схемы. Для создания GDSII обращаются к библиотекам компонентов, цифровых ячеек и IO-ячеек. Далее запускается DRC/LVS, после проверок разработка кристалла завершена. Представленная ниже блок-схема – значительное упрощение маршрута проектирования (рис. 6.10). В больших компаниях петлей обратных связей намного больше, также нужно учитывать необходимость ручных доработок.

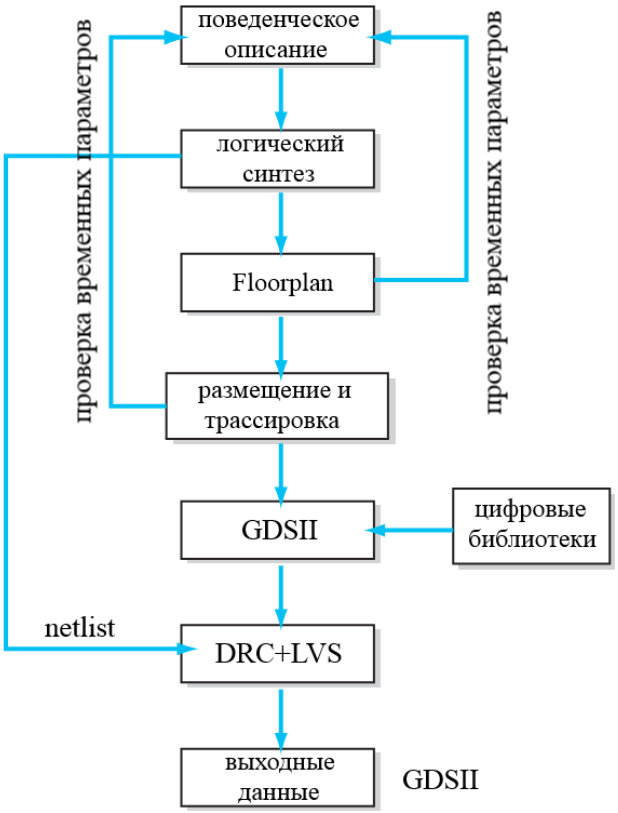


Рис. 6.10. Маршрут проектирования цифровой ИМС

6.11. Вопросы для самопроверки

1. Какие языки высокого уровня используются для описания цифровых устройств? Дайте краткую характеристику каждому из перечисленных языков.
2. Опишите структуру процесса разработки цифровых устройств. Поясните назначение каждого этапа проектирования.
3. Поясните последовательность этапов синтеза топологии цифрового устройства.
4. Что представляет собой оптимизация топологии цифрового устройства? Какие виды оптимизации существуют?
5. Что такое дерево синхронизации? Для чего выполняется его оптимизация?
6. Перечислите основные этапы физической верификации топологии цифровых устройств.

7. ТЕХНИКА РАБОТЫ СО СТАНДАРТНЫМИ ЯЧЕЙКАМИ

7.1. Введение

Современная методология проектирования топологи цифровых устройств предусматривает, автоматизированный синтез топологии на основе имеющихся входных данных. Под входными данными понимается синтезированная схема (нетлист), построенная на основе RTL-описания и библиотеки технологического процесса. Так же в качестве входных данных для синтеза топологии выступают топологические примитивы, т.е. ячейки, на основе которых построен нетлист, и набор различных технологических файлов. Автоматизированное проектирование цифровых схем расширяет требования к базовому набору библиотек. В результате формируется, так называемый, цифровой Design Kit, или библиотека, предназначенная для синтеза. Для автоматизации расстановки и подключения компонентов необходимо выполнять определенные правила разработки цифровых ячеек. Представьте кубики LEGO, они имеют соединения в одних и тех же местах, расположенных на определенном расстоянии. Все блоки стыкуются между собой, все они выполнены в единой сетке. Причина того, что все блоки подходят друг другу в том, что все они кратной длины, ширины, высоты, соединительные шипы расположены с определенным шагом. Ячейки цифровой библиотеки должны быть спроектированы по тем же правилам, что и блоки LEGO.

Из данной главы вы узнаете:

- ✓ *Как стандартизация сетки влияет на цифровую топологию*
- ✓ *Правила разработки цифровых элементов*
- ✓ *Какой подход лучше: переменная высота, фиксированная ширина или фиксированная высота, переменная ширина*
- ✓ *Как выполнить трассировку цифровой ИМС при малом количестве доступных уровней металла*
- ✓ *Что такое "половинные" правила*

7.2. Состав цифровой библиотеки

Основные составляющие библиотеки цифровых элементов:

1. базовые логические элементы (и-не, или-не, защелки, триггеры, мультиплексоры, регистры и др.);

2. lib-файл, в котором содержатся полные сведения о каждом библиотечном элементе (размеры, задержки при различных температурах, напряжении питания, нагрузках). Файл содержит название ячейки, список входов и выходов, описание электрических параметров входов и выходов (входные сопротивления, емкости нагрузки и др.), временные характеристики в виде таблицы коэффициентов в зависимости от нагрузки и температуры;

3. lef-файл технологии содержит общие технологические данные, такие как: сетка размещения, сетка трассировки, параметры слоев металла, ширины шин трассировки, конструкция контактов;

4. lef-файл библиотечных элементов включает в себя название ячейки, классовую принадлежность ячейки (например, ячейка ядра или IO-ячейка), размеры, симметрия относительно осей X и Y, описание выводов ячейки: название, тип (input, output, inout), коэффициент антенного фактора, координаты вывода;

5. tbl-файл, содержит сведения о паразитных элементах слоев трассировки, формирующихся в зависимости от взаимного расположения и конструкции топологических слоев. После окончания первичного синтеза топологии оцениваются временные параметры топологии, базирующиеся на информации, получаемой из tbl-файла и списка задержек для каждой ячейки, извлекаемой из lib-файла;

6. v-файл библиотечных элементов представляет собой Verilog-описания библиотечных ячеек. Файл содержит название ячейки, список входов и выходов, функциональное описание;

7. управляющие файлы физической верификации.

Перечисленные файлы содержат все основные сведения о составе библиотеки и правила размещения, трассировки и оптимизации топологии.

Базовые библиотечные элементы являются структурной основой синтезируемой электрической схемы и топологии устройства. Для каждого библиотечного элемента существует несколько представлений: layout, schematic, symbol, functional и др. Обычно в составе типовой библиотеки присутствуют несколько базовых наборов библиотечных элементов. Кроме базовой библиотеки может присутствовать, например, библиотека для проектирования схем с низким энергопотреблением или с высоким быстродействием, либо с пониженным или повышенным напряжением питания. Для проектирования конкретной схемы выбирается тот вариант, который отвечает требованиям технического задания.

Библиотека базовых элементов содержит в своем составе:

- библиотечные элементы логики и комбинационной логики, такие как: одно-, двух- и многовходовые NOR, OR, AND, NAND, XOR, инверторы, мультиплексоры и многие другие, а также их комбинации;
- библиотечные элементы регистровой логики: защелки, триггеры;
- библиотечные вспомогательные элементы: ячейки заполнения (feeders), ячейки буферных емкостей (decap – *decoupling capacitor*);
- библиотечные ячейки ввода/вывода.

7.3. Стандартизация сетки

Определение размера сетки. Предположим, что минимальная ширина шины первого металла 1 мкм, минимальное расстояние между металлами также 1 мкм, т. о. две параллельные шины занимают 3 мкм ширины. Как видно из рисунка 7.1, расстояние между центрами шин 2 мкм, таким образом, трассировочная сетка составляет 2 мкм. Аналогичная схема применяется для второго и последующего уровней металла. Нужно определить горизонтальную и вертикальную сетки для всего кристалла. Программа-трассировщик может размещать шины только по сетке (рис. 7.2).

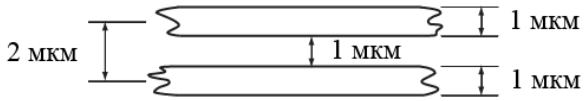


Рис. 7.1. Стандартизация сетки при минимальной ширине шины 1 мкм, мнимальном расстоянии между шинами 1 мкм.

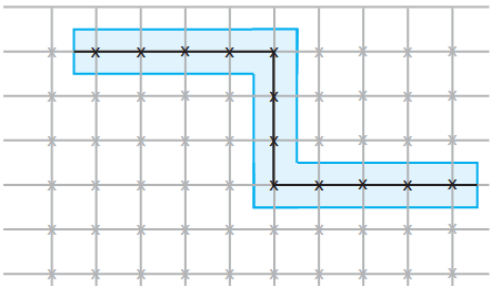


Рис. 7.2. Автоматический трассировщик может располагать шины только по сетке

В современных техпроцессах шаги сетки (*grid pitches*) различны, т.к. металлы и расстояния меду ними для разных слоев отличаются. Если сделать универсальную сетку для всех слоев, то это будет пустой тратой места. Такие трассировщики называются

rule-based routers. Вместо фиксированной сетки, они использует реальные технологические правила.

Направление слоев. Если начать трассировку в одном уровне металла произвольно, то это преградит возможность создания других соединений в дальнейшем. Поэтому необходимо переходить на следующие уровни металла через переходные отверстия via, которые также должны располагаться точно в сетке (рис. 7.3). Начиная произвольно менять уровни металла, потребуется все большее и большее их количество. Как насчет того, чтобы расположить первый металл горизонтально, а второй вертикально (рис. 7.4)? При изменении направления шины нужно поменять уровень металлизации, тогда первый металл никогда не пересечет первый, второй металл не пересечет второй. Есть исключение из данного правила: не стоит менять уровень металла при изгибе шины на один или два шага сетки (рис. 7.5). Смена металла нужна для более комплексной трассировки. При малых изгибах этого не достигнуть, так как эта область блокируется для дальнейшей трассировки, к тому же via – дополнительное сопротивление.

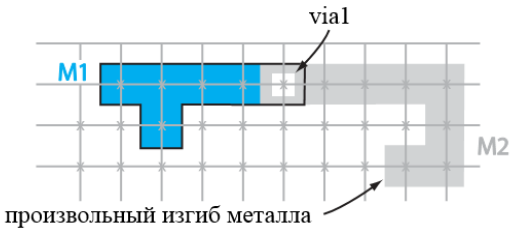


Рис. 7.3. Произвольные скачки между металлами

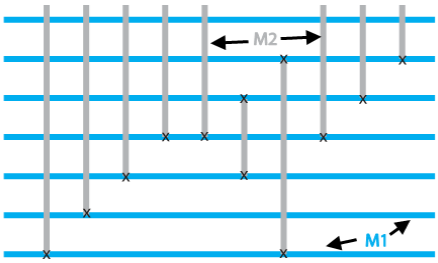


Рис. 7.4. Выбор ориентации металла: первый – горизонтально, второй – вертикально

Рис. 7.5. Не стоит менять уровень металла на небольших изгибах шин

При автоматизированном минтезе топологии проводящим слоям назначаются разрешенные направления трассировки. Как правило, слои металла, имеющие нечетный номер, используются для горизонтальной трассировки, а четные для вертикальной. Слой MET1 обычно используется для построения базовых библиотечных ячеек и локальных межсоединений между ними. Дополнительно слой MET1 может использоваться для создания частей колец питания.

7.4. Правила для библиотек базирующихся на сетке

Все используемые в проекте цифровые элементы должны придерживаться следующих правил:

1. Выравнивание входа и выхода. На рисунке 7.6 представлен стандартный инвертор. Вход – A, выход – Z. Положение входа и выхода нельзя выбирать произвольно, они должны располагаться в одной сетке. При выравнивании необходимо обеспечить наиболее простое соединение между A и Z.

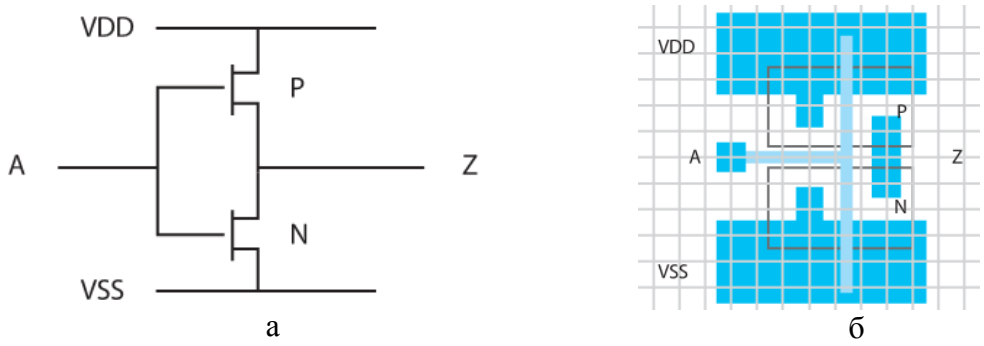


Рис. 7.6. Различные представления инвертора:
а – схемотехническое; б –топологическое

2. Фиксированная высота, переменная ширина. Если каждый элемент библиотеки имеет свою уникальную высоту и шины питания разной ширины, то соединения будут беспорядочными, даже если все элементы выровнены по сетке. На рис. 7.7 представлен пример "блуждающих" рейлов питания.

Таким образом, чтобы выровнять топологию необходимо придерживаться правила фиксированной высоты. Для этого транзисторы с большой шириной, работающие на большую нагрузку, разделяют на маленькие и подгоняют под рейлы. Минимальная высота ячейки обычно определяется размерами транзисторов, сеткой и шириной рейла.

При стыковке элементов рейлы питания автоматически будут выровнены. Поэтому большинство цифровых библиотек построено по принципу фиксированная высота, переменная ширина.

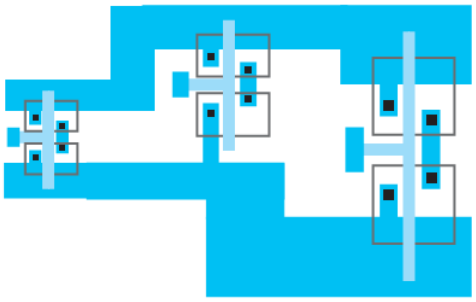


Рис. 7.7. Пример элементов с разной высотй и одинаковой шириной

7.5. Цифровые библиотеки: фиксированная высота, переменная ширина

1. Определение калибра шин. Единичная сетка позволяет провести одну шину. Шины питания обычно делают шириной в 2-, 3- шага сетки (рис. 7.8).

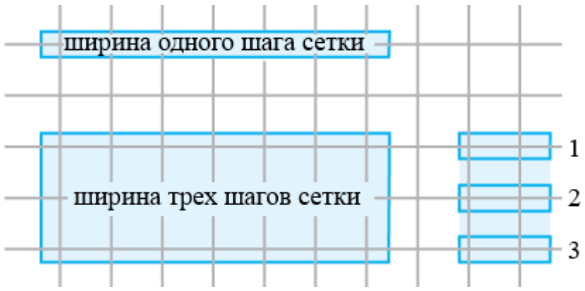


Рис. 7.8. Шина питания в три шага сетки

2. Общий N-карман. Предположим, что нужно расположить 4 цифровые ячейки рядом. Обычно элементы нужно разместить как можно ближе друг к другу. В обычном КМОП процессе заложены правила на большое расстояние между NWELL (n-карманы, в которых располагаются p- МОП транзисторы). В большинстве случаев, в цифровых элементах подложка p- МОП подключена к VDD, поэтому удобно сделать один большой n-карман, это позволит уменьшить площадь, т.к. ограничением расстояния между элементами станут минимальные расстояния между областями диффузии (рис. 7.9, 7.10). Таким образом, ячейки должны быть разработаны так, чтобы при их стыковке расстояние между транзисторами было минимально допустимым.

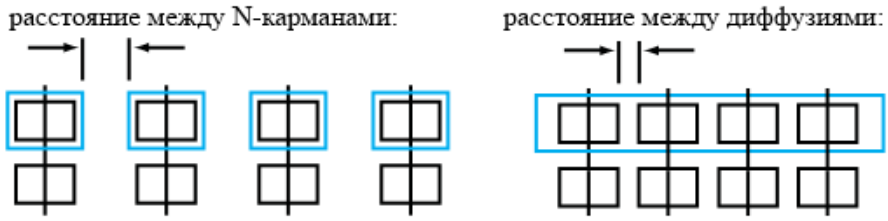


Рис. 7.9. Создание общего N-кармана для p-канальных транзисторов

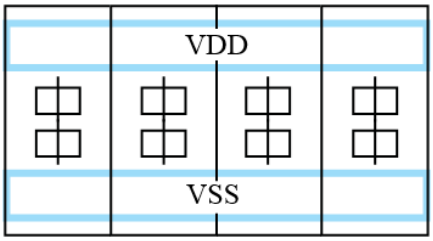


Рис. 7.10. Стыковка элементов с формированием рейлов земли и питания

3. Размер ячейки в половину шага сетки. После стыковки ячеек друг с другом шины питания и полоски NWELL будут автоматически подключены. Но как должны быть расположены транзисторы относительно границы ячейки, чтобы при стыковке между транзисторами было минимально допустимое расстояние? Для этого необходимо выполнять внутренние подключения в сетке, что гарантирует при стыковке, что металлы будут на минимальном расстоянии (рис. 7.11). Ячейки должны стыковаться во всех направлениях, поэтому с каждой стороны – сверху, снизу, слева, справа границы располагают на расстоянии в половину шага сетки.

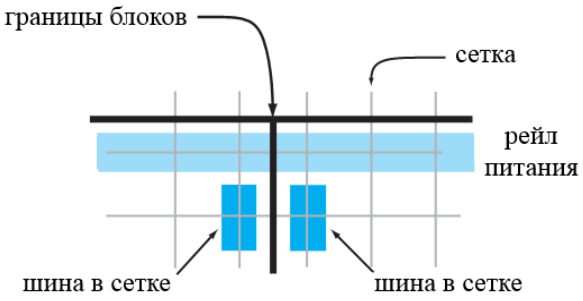


Рис. 7.11. Выбор расположение границы ячейки

4. Половинные правила. Поскольку граница ячейки находится в середине сетки, то действует как минимум одно половинное правило по расположению транзисторов относительно границы ячейки (рис. 7.12). В общий состав слоев добавляется до-

полнительный слой, необходимый синтезатору топологии для определения габаритов ячеек – это слой «boundary». Данный слой содержит любая библиотечная цифровая ячейка. Габариты слоя являются определяющими при размещении библиотечных ячеек и блока. Если ячейка оказывается немного шире и ее граница выходит за половину сетки, то ячейку расширяют на один полный шаг сетки.

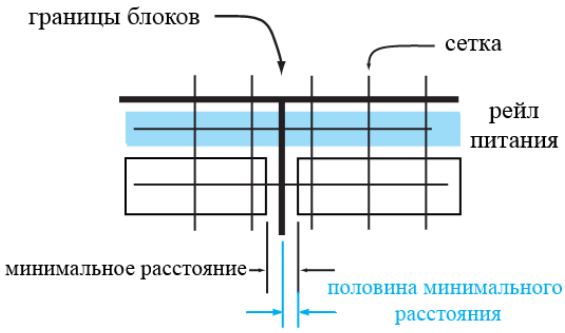


Рис. 7.12. Иллюстрация половинных правил

7.6. Каналы трассировки

Если для трассировки разрешено использовать малое число металлов, то, следует придерживаться некоторых рекомендаций. Если имеется несколько рядов элементов, то один ряд отражают по вертикали и сливают с другим рядом по шине земли или питания. Рейлы питания достаточно подключить первым металлом за границей сборки (рис. 7.13а) или объединить вторым уровнем металла (рис. 7.13б).

Трассировку в ячейках желательно выполнять в первом уровне металле. Для этого нужно с внешней стороны рейлов и между ячейками оставлять зазоры – **каналы трассировки**, они нужны для создания подключений в дальнейшем (рис. 7.14). Таким образом, если есть ограничения в количестве уровней металла, то нужно увеличить размер ячеек на 1-3 дополнительных шага сетки над рейлами, а не сливать отраженные ячейки.

Предположим, что в проекте очень большое число шин и каналов трассировки не хватает для подключения всех сетей. В этом случае есть канальные трассировщики с переменной шириной (*variable-width routing*) (рис. 7.15), создающие дополнительно каналы между ячейками в пределах ряда, а также увеличивающий ширину канала там, где

это необходимо. Варьируя ширину каналов можно получить более компактную топологию.

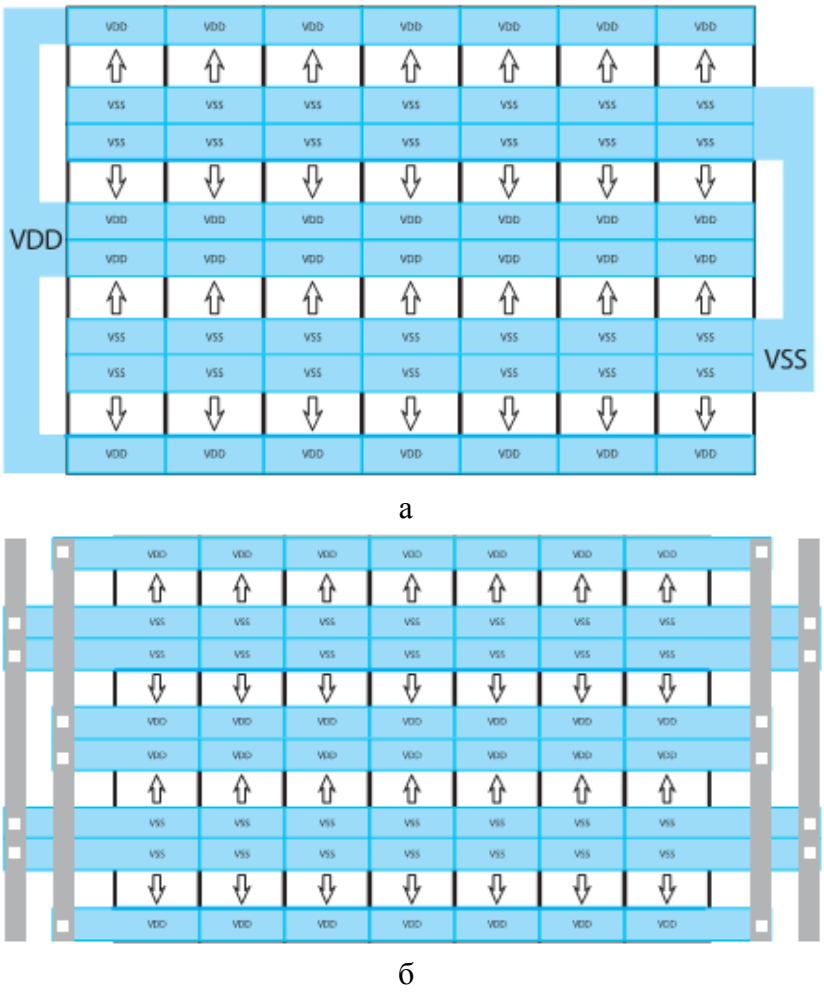


Рис. 7.13. Стыковка рядов ячеек и объединение рейлов:
а – первым уровнем металла; б – вторым уровнем металла

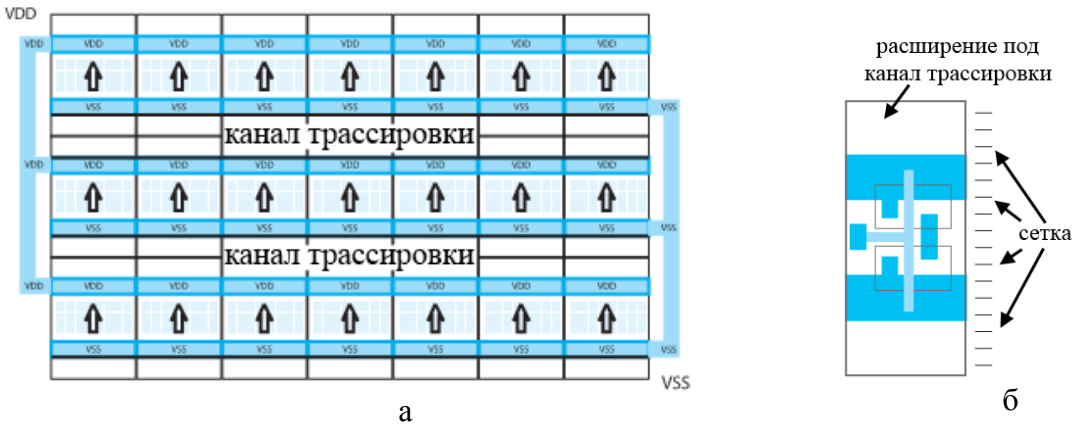


Рис. 7.14. Выделение мест для каналов трассировки (а) и инвертор с расширением под канал трассировки (б)

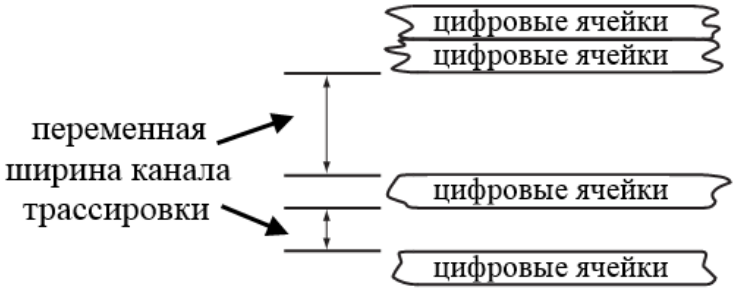


Рис. 7.15. Переменная ширина каналов трассировки

7.7. Антенные правила

При подключении входа одной цифровой ячейки к выходу другой короткой шиной первого металла, затворы первой ячейки оказываются подключенными к диодам сток-подложка. Затворы транзисторов, которые не подключены к диодам необходимо защищать дополнительными антенными NAC- диодами (*Net Area Check*) (рис. 7.16). Антенный эффект – это процесс накопления заряда на металлических шинах в процессе плазмохимического травления. Если площадь шины, подключенной к затвору транзистора большая, то между затвором и подложкой транзистора может возникнуть разность потенциалов, достаточная для пробоя подзатворного диэлектрика. Антенные диоды, подключенные между затвором и подложкой транзистора, открываются, когда разность потенциалов между катодом и анодом достигает порогового значения. Для выявления мест возможного пробоя используются специализированные антенные правила, которые обычно проверяются одновременно с DRC. Цифровые библиотеки дополняют ячейками, содержащими антенные диоды для автоматизации их расстановки при синтезе топологии.

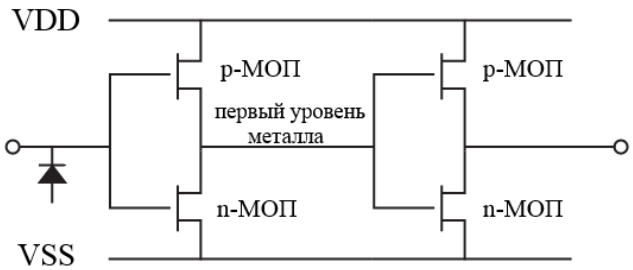


Рис. 7.16. Защита входных затворов антенными диодами

7.8. Стандартизация IO-ячеек

Как и цифровые ячейки, ячейки ввод-вывода также подчиняются правилу сетки. Пример простой IO ячейки или пада (*input-output*) представлен на рис. 7.17а. Нижние два рейла обычно являются питанием входного драйвера, верхние два рейла – ESD- шины. Единственно отличие процесса синтеза топологии, содержащей пады, от синтеза топологии цифровых блоков заключается в том, что IO-ячейки можно располагать кольцом, который называют IO-ring (рис. 7.17б).

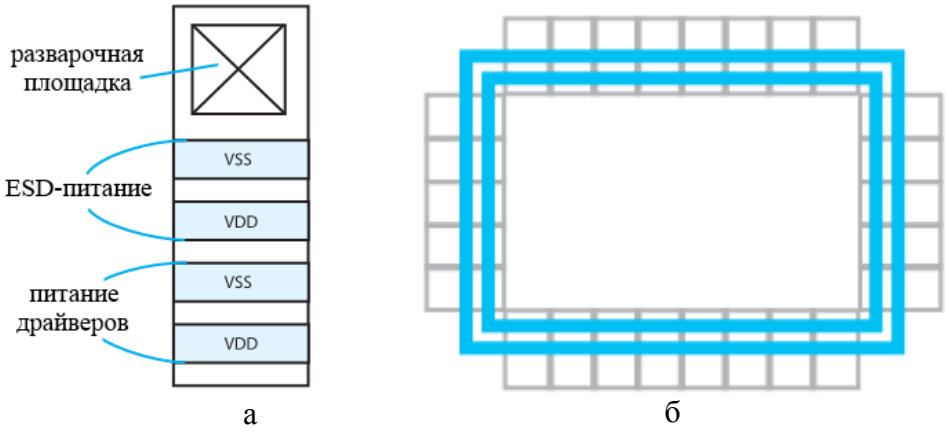


Рис. 7.17. IO-ячейка (а) и IO-ring (б)

Все блоки ввода–вывода обычно максимально удаляются от остальных частей схемы, это связано с наличием мощных переключающихся каскадов в структуре блоков, способных навести существенные помехи на остальную часть схемы. Кроме того каждая ячейка ввода–вывода должна быть обеспечена электростатической защитой (*electro static discharge - ESD*). Электростатические разряды – одна из причин отказов интегральных схем. Заряды возникают в результате явлений, приводящих к появлению разностей электрических потенциалов между отдельными элементами оборудования, которые соприкасаются со схемами, между человеком и изделием или оборудованием и т.п. Внезапные разряды приводят к протеканию импульсов токов, как правило, очень коротких по времени, но имеющих большую амплитуду и способных полностью или частично повредить внутреннюю структуру схемы. Повышение быстродействия, снижение потребляемой мощности, уменьшение геометрических разрядов элементов полупроводниковых схем делают их еще более чувствительными к воздействию электрических полей,

и в особенности электростатического разряда. На рисунке 7.18 представлена типовая схема защиты входных каскадов цифровых схем.

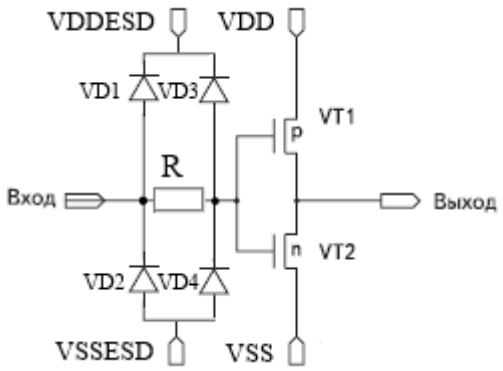


Рис. 7.18. Типовая защита затворов транзисторов организованная на диодах первичной (VD1, VD2), вторичной (VD3, VD4) ESD-защиты и последовательного сопротивления R

Более сложные схемы защиты детектируют электростатический разряд на периферийных шинах питания и земли и замыкают их между собой (так называемые clamp-структуры). В качестве детектора электростатического разряда и элемента, запускающего включение схемы защиты, выступает емкость. При превышении порога напряжения широкий выходной транзистор открывается, замыкая выводы питания и земли, что приводит к уменьшению разности потенциалов между выводами схемы. Для надежной защиты схемы ячейки электростатической защиты располагаются по периферии кристалла с определенным шагом, а также в углах кристалла.

7.9. Вопросы для самопроверки

1. Что содержит библиотека цифровых элементов?
2. Поясните назначение библиотечных файлов lef, tbl, lib.
3. Для чего проводится стандартизация цифровых ячеек?
4. Какими способами обеспечивается максимальная плотность элементов при автоматическом синтезе топологии?
5. Опишите особенности синтеза топологии при большом и малом числе доступных уровней металла.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Рабаи Ж. М. Цифровые интегральные схемы. Методология проектирования: пер. с англ. / Б. Уилкинсон. М.: ИД «Вильямс», 2004. – 894 с.
2. Поляков А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. – М.: СОЛОН-Пресс, 2003. – 320 стр.
3. Ракитин В. В. Интегральные схемы на КМОП-транзисторах / В. В. Ракитин. – М.: 2007. – 307 с.
4. Saint C. IC Mask Design. Essential layout techniques / C. Saint, J. Saint. – New York, USA: McGraw-Hill, 2002. – 394 p.
5. Kaeslin H. Digital Integrated Circuit Design / H. Kaeslin. – New York: Cambridge University Press, 2008. – 845 p.
6. Сафонов И.А. Системное проектирование 3D изделий. Методические указания к лабораторным работам / И. А. Сафонов. – Воронеж: ВГТУ, 2010.

ЦИФРОВЫЕ ОБРАЗОВАТЕЛЬНЫЕ РЕСУРСЫ

1. www.cadence .com – сайт производителя ПО САПР Cadence
2. www.mentor.com – сайт производителя ПО САПР Mentor Graphics
3. www.electronics.ru – сайт журнала «Электроника НТБ»
4. www.russianelectronics.ru – ресурс «Время электроники»
5. www.kit-e.ru – сайт журнала «Компоненты и технологии»

У ч е б н о е и з д а н и е

ВВЕДЕНИЕ В СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ИНТЕГРАЛЬНЫХ МИКРОСХЕМ

Часть I

Учебно-методическое пособие

Составители:
Тучин Андрей Витальевич,
Бормонтов Евгений Николаевич,
Пономарев Константин Геннадьевич

Издано в авторской редакции

Подписано в печать 29.06.2017. Формат 60×84/16
Уч.-изд. л. 5,5. Усл. п. л. 6,5. Тираж 25 экз. Заказ 389

Издательский дом ВГУ
394018 Воронеж, пл. им. Ленина, 10

Отпечатано с готового оригинал-макета
в типографии Издательского дома ВГУ
394018 Воронеж, ул. Пушкинская, 3