Областное государственное бюджетное профессиональное образовательное

учреждение «Смоленская академия профессионального образования»

Теоретический материал

для выполнения самостоятельной работы студентов

по дисциплине САПР электрических схем и печатных плат

для студентов специальности

12.02.03 (200111) Радиоэлектронные приборные устройства

Смоленск 2014 г.

СОДЕРЖАНИЕ

Раздел 1. Методика и средства автоматизированного проектирования

Тема 1.1. Введение в автоматизированное проектирование

Тема 1.2. Аппаратура рабочих мест в автоматизированных системах проектирования и управления

Тема 1.3. Математическое обеспечение анализа проектных решений

Тема 1.4. Математическое обеспечение синтеза проектных решений

Тема 1.5. Системные среды САПР

Раздел 2. Инструментарий проектировщика

Тема 2.1. Методики проектирования автоматизированных систем

Тема 2.2. Основные сведения о языке VHDL

Тема 2.3. Назначение и состав системных сред САПР

Тема 2.4. Проектирование радиоэлектронных средств САПР в P-CAD

**Раздел 1. Методика и средства автоматизированного проектирования**

**Тема 1.1. Введение в автоматизированное проектирование**

Проектирование это комплекс работ по исследованию расчетами конструированию нового изделия или нового процесса В основе проектирования лежит первичное описание или техническое задание .

В условиях жесткой конкуренции внедрение новых информационных технологий позволяет:

- повысить техническое качество проектов;

- применить новые технические решения;

- сократить сроки проектирования;

- эффективнее реагировать на выдвигаемые заказчиком требования;

- оперативно и качественно выполнять необходимые изменения и корректировку проектов;

- выдавать заказчику проектно–сметную документацию в современных цифровых форматах;

- насыщать проектную документацию дополнительной атрибутивной информацией используемой в эксплуатации;

- повысить эффективность управления проектированием.

Различают следующие виды проектирования неавтоматизированное проектирование и автоматизированное проектирование.

Процесс проектирования осуществляемый человеком вручную и называют неавтоматизированным проектированием. Проектирование при котором все проектные решения или их часть получают путем взаимодействия человека с ЭВМ называют автоматизированным проектированием.

*Стадии разработки проекта можно представить следующим образом:*

Этапы выполнения работ.

Техническое предложение.

Подбор материалов.

Разработка технического предложения с присвоением документам литеры «П».

Рассмотрение и утверждение технического предложения

Эскизный проект

Разработка эскизного проекта с присвоением документам литеры «Э».

Изготовление и испытание материальных макетов (при необходимости) и (или) разработка, анализ электронных макетов (при необходимости)

Рассмотрение и утверждение эскизного проекта.

## Технический проект

Разработка технического проекта с присвоением документам литеры «Т».

Изготовление и испытание материальных материальных макетов (при необходимости) и (или) разработка, анализ электронных макетов (при необходимости).

Рассмотрение и утверждение технического проекта.

Рабочая конструкторская документация:

а) опытного образца (опытной партии) изделия, предназначенного для серийного (массового) или единичного производства (кроме разового изготовления)

Разработка конструкторской документации, предназначенной для изготовления и испытания опытного образца (опытной партии), без присвоения литеры.

Изготовление и предварительные испытания опытного образца (опытной партии).

Корректировка конструкторской документации по результатам изготовления и предварительных испытаний опытного образца (опытной партии) с присвоением документам литеры «О».

Приемочные испытания опытного образца (опытной партии).

Корректировка конструкторской документации по результатам приемочных испытаний опытного образца (опытной партии) с присвоением документам литеры «О1».

Для изделия, разрабатываемого по заказу Министерства обороны, при необходимости, - повторное изготовление и испытания опытного образца (опытной партии) по документации с литерой «О1» и корректировка конструкторских документов с присвоением им литеры «О2».

б) серийного (массового) производства

Изготовление и испытание установочной серии по документации с литерой «О1» (или «О2»).

Корректировка конструкторской документации по результатам изготовления и испытания установочной серии, а также оснащения технологического процесса изготовления изделия, с присвоением конструкторским документам литеры «А».

Для изделия, разрабатываемого по заказу Министерства обороны, при необходимости, - изготовление и испытание головной (контрольной) серии по документации с литерой «А» и соответствующая корректировка документов с присвоением им литеры «Б»



**Системы автоматизированного проектирования и их место среди других автоматизированных систем**

|  |
| --- |
| Российские предприятия используют в основном чертежные 2D-системы (AutoCAD, T-flex, КОМПАС и т.д.) и недорогие программы для механообработки (AlphaCAM, Техтран, ТИГРАС, MasterCAM и т.д.). Программные пакеты объемного моделирования из-за их дороговизны, повышенных требований к квалификации персонала и сложности освоения применяются намного реже, хотя задач, для решения которых необходимы именно 3D-системы, становится все больше.Возможно, когда-нибудь 3D-системы полностью вытеснят 2D с рынка CAD/CAM-продуктов. Но и на сегодня, и на многие годы вперед выполнение чертежей для производства является насущной необходимостью. Следовательно, чертежным CAD-системам, пусть и теснимым постепенно системами объемного [3D-моделирования](http://www.autodesk.ru/products/autodesk-autocad/overview), суждена еще очень долгая жизнь. САПР двумерного проектирования — «2D-3D Легкие — Нижний уровень». Эти САПР служат для выполнения почти всех работ с двумерными чертежами и имеют ограниченный набор функций по трехмерному моделированию. С помощью этих систем выполняются порядка 90% всех работ по проектированию. Хотя имеющиеся ограничения делают их не всегда довольно удобными. Область их работы — создание чертежей отдельных деталей и сборок. Платой за возросшие возможности является усложнение интерфейса и меньшее удобство в работе. Характерные представители таких САПР — AutoCAD, CADdy, CADMECH Desktop, MasterCAM, T-FlexCAD, OmniCAD, Компас-График.1. САПР объемного моделирования «3D — Средний уровень»

По своим возможностям они полностью охватывают САПР «легкого веса», а также позволяют работать со сборками, по некоторым параметрам они уже не уступают тяжелым САПР, а в удобстве работы даже превосходят. Обязательным условием является наличие функции обмена данными (или интеграции). Это не просто программы, а программные комплексы, в частности, SolidWorks SolidEdge, Cimatron, Form-Z, Autodesk Inventor, CAD SolidMaster, и все еще продолжающий развиваться, Mechanical Desktop, DesignSpace.1. САПР объемного моделирования «3D Тяжелые — Верхний уровень»

Эти системы применяются для решения наиболее трудоемких задач - моделирования поведения сложных механических систем в реальном масштабе времени, оптимизирующих расчетов с визуализацией результатов, расчетов температурных полей и теплообмена и т.д. Обычно в состав системы входят как чисто графические, так и модули для проведения расчетов и моделирования, постпроцессоры для станков с ЧПУ. К сожалению, эти самые мощные САПР наиболее громоздки и сложны в работе, а также имеют значительную стоимость. Примерами «тяжелых» САПР могут служить такие продукты, как ADAMS, ANSYS, CATIA, EUCLID3, Pro/ENGINEER, UniGraphics.САПР «тяжелого» уровня не оптимальны для выпуска и корректировки конструкторской документации, которая по-прежнему составляет максимальную долю затрат на проектирование изделия. По мнению экспертов, количество рабочих мест таких САПР должно составлять приблизительно 5-10% от общего количества рабочих мест. Эта цифра подтверждается примерами наиболее успешных внедрений САПР на отечественных предприятиях, например, в САПР ЦКБ МТ «Рубин».В настоящее время наиболее распространены следующие типы САПР для автоматизированного проектирования в машиностроении:* системы автоматизированного черчения CADD (Computer Aided Design and Drafting, »);
* системы автоматизированного построения технологических процессов CAM (Computer Aided Mechanical Process, «компьютерная помощь в механообработке»);
* системы автоматизации инженерных расчетов CAE (Computer Aided Engineering, «компьютерная помощь в инженерных расчетах) и трехмерного моделирования» (в том числе геометрического);
* системы САПР для подготовки данных для станков с ЧПУ (постпроцессоры);
* специализированные САПР (например для проектирования коробок передач);
* интегрированные системы, включающие в себя несколько вышеперечисленных.

В зависимости от области применения требования к САПР сильно отличаются, но можно выделить основные:1. Система должна быть открытой, т. е. пользователь должен иметь возможность настраивать и надстраивать систему в зависимости от своих нужд. Например, пользователь может подключать свои программные модули, написанные на языках программирования высокого уровня.
2. Система должна работать со стандартными протоколами обмена и хранения информации.Обязательна поддержка ГОСТ и ЕСКД (для конструкторских САПР). Крайне желательно наличие функций моделирования и параметрического проектирования.
3. Желательно, чтобы система функционировала на различных аппаратных и программных платформах.
4. Системой должна поддерживаться работа над проектом в многопользовательском режиме.
5. Необходима интеграция САПР в единую систему электронного документооборота и архива предприятия.
 |

**Тема 1.2. Аппаратура рабочих мест в автоматизированных системах проектирования и управления**

## Классификация САПР

ГОСТ 23501.108-85 устанавливает следующие признаки классификации САПР:

* тип/разновидность и сложность объекта проектирования
* уровень и комплексность автоматизации проектирования
* характер и количество выпускаемых документов
* количество уровней в структуре технического обеспечения



Рисунок 1. Классификация САПР

В области классификации САПР используется ряд устоявшихся англоязычных терминов, применяемых для классификации программных приложений и средств автоматизации САПР по отраслевому и целевому назначению.

**По отраслевому назначению:**

* MCAD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) mechanical computer-aided design) — автоматизированное проектирование механических устройств. Это машиностроительные САПР, применяются в автомобилестроении, судостроении, авиакосмической промышленности, производстве товаров народного потребления, включают в себя разработку деталей и сборок (механизмов) с использованием параметрического проектирования на основе конструктивных элементов, технологий поверхностного и объемного моделирования ([SolidWorks](https://ru.wikipedia.org/wiki/SolidWorks%22%20%5Co%20%22SolidWorks), [Autodesk Inventor](https://ru.wikipedia.org/wiki/Autodesk_Inventor), [КОМПАС](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D1%81_%28%D0%A1%D0%90%D0%9F%D0%A0%29), [CATIA](https://ru.wikipedia.org/wiki/CATIA));
* [EDA](https://ru.wikipedia.org/wiki/EDA) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) electronic design automation) или ECAD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) electronic computer-aided design) — САПР [электронных устройств](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0#.D0.AD.D0.BB.D0.B5.D0.BA.D1.82.D1.80.D0.BE.D0.BD.D0.BD.D1.8B.D0.B5_.D1.83.D1.81.D1.82.D1.80.D0.BE.D0.B9.D1.81.D1.82.D0.B2.D0.B0), [радиоэлектронных средств](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%BE%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0), [интегральных схем](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0), [печатных плат](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%87%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0) и т. п., ([Altium Designer](https://ru.wikipedia.org/wiki/Altium_Designer%22%20%5Co%20%22Altium%20Designer), [OrCAD](https://ru.wikipedia.org/wiki/OrCAD));
* AEC CAD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) architecture, engineering and construction computer-aided design) или CAAD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided architectural design) — САПР в области архитектуры и строительства. Используются для проектирования зданий, промышленных объектов, дорог, мостов и проч. ([Autodesk Architectural Desktop](https://ru.wikipedia.org/w/index.php?title=Autodesk_Architectural_Desktop&action=edit&redlink=1), [AutoCAD Revit Architecture Suite](https://ru.wikipedia.org/w/index.php?title=AutoCAD_Revit_Architecture_Suite&action=edit&redlink=1), Bentley MicroStation, Bentley AECOsim Building Designer, [Piranesi](https://ru.wikipedia.org/wiki/Piranesi), [ArchiCAD](https://ru.wikipedia.org/wiki/ArchiCAD)).

#### По целевому назначению

По целевому назначению различают САПР или подсистемы САПР, которые обеспечивают различные аспекты проектирования

* CAD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided design/drafting) — средства автоматизированного проектирования, в контексте указанной классификации термин обозначает средства САПР, предназначенные для автоматизации двумерного и/или трехмерного геометрического проектирования, создания конструкторской и/или технологической документации, и САПР общего назначения.
	+ CADD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided design and drafting) — проектирование и создание чертежей.
	+ CAGD ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided geometric design) — геометрическое моделирование.
* [CAE](https://ru.wikipedia.org/wiki/Computer-aided_engineering) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided engineering) — средства автоматизации инженерных расчётов, анализа и симуляции физических процессов, осуществляют динамическое моделирование, проверку и оптимизацию изделий.
	+ CAA ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided analysis) — подкласс средств CAE, используемых для компьютерного анализа.
* [CAM](https://ru.wikipedia.org/wiki/CAM) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided manufacturing) — средства технологической подготовки производства изделий, обеспечивают автоматизацию программирования и управления оборудования с [ЧПУ](https://ru.wikipedia.org/wiki/%D0%A7%D0%9F%D0%A3) или ГАПС (Гибких автоматизированных производственных систем). Русским аналогом термина является [АСТПП](https://ru.wikipedia.org/wiki/%D0%90%D0%A1%D0%A2%D0%9F%D0%9F) — автоматизированная система технологической подготовки производства.
* [CAPP](https://ru.wikipedia.org/wiki/CAPP) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) computer-aided process planning) — средства автоматизации планирования технологических процессов, применяемые на стыке систем CAD и CAM.

Многие системы автоматизированного проектирования совмещают в себе решение задач, относящихся к различным аспектам проектирования CAD/CAM, CAD/CAE, CAD/CAE/CAM. Такие системы называют комплексными, или интегрированными.

С помощью CAD-средств создаётся [геометрическая модель](https://ru.wikipedia.org/w/index.php?title=%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C&action=edit&redlink=1) изделия, которая используется в качестве входных данных в системах CAM и на основе которой в системах CAE формируется требуемая для инженерного анализа модель исследуемого процесса.

**Функции сетевого программного обеспечения**

[**Примеры серверов**](http://bigor.bmstu.ru/?met/?doc=103_Computers/1038.mod/?cou=140_CADedu/CAD.cou/?bck=103_Computers/1038.mod)

Серверы подразделяются по выполняемым функциям на серверы файловые, баз данных, приложений и т.п., по масштабом сети, в которой они используются, на серверы локальные (для подсетей подразделений или отделов предприятия) и корпоративные, по производительности — на серверы высокого (hi-end), среднего и младшего уровней.

Для серверов характерны следующие особенности:

* повышенные требования к производительности, поддерживаемые применением многопроцессорности и многоядерности, развитием технологий ввода/вывода данных, увеличением пропускной способности шин;
* повышенные требования к отказоустойчивости, реализуемые, в частности, с помощью технологии дисковых массивов RAID;
* возможность удаленного управления.

Для серверов типично (по состоянию на конец 2006 г.) применение архитектуры SMP, двух- и четырехядерных процессоров, интерфейсов Gigabit Ethernet, шин PCI Express и Infiniband.

С помощью Infiniband удается строить так называемые *серверы-лезвия* (blade servers), отличительной особенностью которых является специфическое конструктивное исполнение. На платах серверов, размещаемых в корпусе высотой в несколько "юнитов" ("юнит" или величина равна 1,75 дюйма), располагается несколько процессоров и оперативная память. Эта плата вставляется в отдельный паз в корпусе, в котором умещаются еще десятки подобных плат. В одну стойку устанавливается несколько серверов, что обеспечивает существенное увеличение вычислительной мощности на сравнительно малой площади В отличие от традиционных систем, построенных по принципу "один корпус — одна системная плата", в серверах-лезвиях несколько системных плат совместно используют общий корпус, блок питания, общие вентиляторы и кабельные соединения.

Совместное использование компонентов — корпуса, соединительной платы, источника питания и вентиляторов — обеспечивает значительную экономическую эффективность аппаратного обеспечения. Меньшее число компонентов обеспечивает повышенную надежность серверов, а плотность их расположения позволяет значительно упростить их обслуживание.

Для связи серверов друг с другом и с внешними устройствами, в том числе с внешней памятью, вместо общей шины используется коммутирующая матрица и последовательный интерфейс Infiniband. В результате получается компактная конструкция из серверных плат с конструктивно отделенными дисковой памятью, устройствами ввода/вывода, источниками питания, достигается уменьшение потребления электроэнергии, тепловыделения, расстояний между серверными платами по сравнению с аналогичными характеристиками обычных кластеров.

Как правило, серверы корпоративного уровня работают под управлением операционных систем UNIX, Linux, а среднего уровня - также Microsoft Windows Server.

Пример сервера корпоративного уровня — IBM System p5 590 поддерживает до 32 процессоров POWER5+ с тактовой частотой 2,1 ГГц, оперативную память объемом до 1 Tбайт и может оснащаться дисковой подсистемой емкостью 18,7 Tбайт.

Примерами серверов среднего уровня могут служить серверы:

* IBM System x3755 (на процессорах AMD Opteron) и System x3950 (на процессорах Intel Xeon) с емкостью оперативной памяти в 128 Гбайт;
* HP ProLiant DL380 G5 с установкой двух четырехъядерных процессоров Intel Xeon, емкостью ОЗУ 32 Гбайт и дисковой памяти 576 Гбайт;
* Sun Fire E2900 на базе двухъядерных 64-разрядных процессоров Sun UltraSPARC IV+, имеет 192 Гбайт оперативной и 292 Гбайт дисковой памяти;
* Dell PowerEdge 6800 на четырех двухъядерных процессорах Xeon MP.

**Тема 1.3. Математическое обеспечение анализа проектных решений**

Совокупность однотипных компонентов образует *средство обеспечения* САПР. Выделяют следующие виды обеспечения САПР:

* [*Техническое обеспечение*](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (ТО) — совокупность связанных и взаимодействующих технических средств ([ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%92%D0%9C), [периферийные устройства](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B8%D1%84%D0%B5%D1%80%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), [сетевое оборудование](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B5_%D0%BE%D0%B1%D0%BE%D1%80%D1%83%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), линии связи, измерительные средства).
* *Математическое обеспечение* (МО), объединяющее математические методы, [модели](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) и [алгоритмы](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), используемые для решения задач автоматизированного проектирования. По назначению и способам реализации делят на две части:
	+ математические методы и построенные на них математические модели;
	+ формализованное описание технологии автоматизированного проектирования.
* *Программное обеспечение* (ПО). Подразделяется на *общесистемное* и *прикладное*:
	+ [*прикладное ПО*](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) 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 (модуль проектирования трубопроводов, пакет схемотехнического моделирования, [геометрический решатель САПР](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%80%D0%B5%D1%88%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_%D0%A1%D0%90%D0%9F%D0%A0)).
	+ [*общесистемное ПО*](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) предназначено для управления компонентами *технического обеспечения* и обеспечения функционирования *прикладных программ*. Примером компонента *общесистемного ПО* является [операционная система](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).
* *Информационное обеспечение* (ИО) — 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 ИО САПР — [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).
* *Лингвистическое обеспечение* (ЛО) — 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, методы сжатия и развертывания.
	+ В лингвистическом обеспечении выделяют класс различного типа [языков проектирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и моделирования ([VHDL](https://ru.wikipedia.org/wiki/VHDL), [VERILOG](https://ru.wikipedia.org/wiki/Verilog), [UML](https://ru.wikipedia.org/wiki/UML), [GPSS](https://ru.wikipedia.org/wiki/GPSS)).
* *Методическое обеспечение* (МетО) — 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](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B5%D0%B7) и их составных частей, различные методики проектирования. Иногда к МетО относят также *МО* и *ЛО*.
* *Организационное обеспечение* (ОО) — совокупность документов, определяющих состав проектной организации, связь между подразделениями, организационную структуру объекта и системы автоматизации, деятельность в условиях функционирования системы, форму представления результатов проектирования… В ОО входят [штатные расписания](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5), [должностные инструкции](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BB%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F), правила эксплуатации, приказы, положения и т. п.

В САПР как проектируемой системе выделяют также эргономическое и правовое обеспечения.

* *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.

**Компоненты математического обеспечения**

[**Классификация методов математического программирования**](http://bigor.bmstu.ru/?met/?doc=120_Opt/5003.mod/?cou=140_CADedu/CAD.cou/?bck=120_Opt/5003.mod)

Основными методами оптимизации в САПР являются поисковые методы. Поисковые методы основаны на пошаговом изменении управляемых параметров

где в большинстве методов приращение вектора управляемых параметров вычисляется по формуле

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\5 Математическое обеспечение синтеза пректных решений\Классификация методов математического программирования_files\saved_resource(12) |  (1) |

Здесь — значение вектора управляемых параметров на -м шаге; — шаг; а — направление поиска. Следовательно, если выполняются условия сходимости, то реализуется пошаговое (итерационное) приближение к экстремуму.

Методы оптимизации классифицируют по ряду признаков.

В зависимости от числа управляемых параметров различают методы одномерной и многомерной оптимизации. В методах *одномерной оптимизации* управляемый параметр единственный, во вторых размер вектора не менее двух. Реальные задачи в САПР многомерны, методы одномерной оптимизации играют вспомогательную роль на отдельных этапах многомерного поиска.

Различают методы условной и безусловной оптимизации по наличию или отсутствию ограничений. Для реальных задач характерно наличие ограничений, однако методы *безусловной оптимизации* также представляют интерес, поскольку задачи *условной оптимизации* с помощью специальных методов могут быть сведены к задачам без ограничений.

В зависимости от числа экстремумов различают задачи одно- и многоэкстремальные. Если метод ориентирован на определение какого-либо локального экстремума, то такой метод относится к локальным методам. Его называют методом *локального поиска*. Если же результатом является глобальный экстремум, то метод называют методом *глобального поиска*. Удовлетворительные по вычислительной эффективности *методы глобального поиска* для общего случая отсутствуют и потому на практике в САПР используют методы поиска локальных экстремумов.

Наконец, в зависимости от того, используются при поиске производные целевой функции по управляемым параметрам или нет, различают методы нескольких порядков. Если производные не используются, то имеет место метод нулевого порядка, если используются первые или вторые производные, то соответственно метод первого или второго порядка. Методы первого порядка называют также градиентными, поскольку вектор первых производных по есть градиент целевой функции


Конкретные методы определяются следующими факторами:

1. способом вычисления направления поиска в формуле (1);
2. способом выбора шага ;
3. способом определения окончания поиска.

Определяющим фактором является первый из перечисленных в этом списке.

Шаг может быть или постоянным, или выбираться исходя из одномерной оптимизации — поиска минимума целевой функции в выбранном направлении . В последнем случае шаг будем называть оптимальным.

Окончание поиска обычно осуществляют по правилу: если на протяжении подряд идущих шагов траектория поиска остается в малой -окрестности текущей точки поиска , то поиск следует прекратить, следовательно,. условие окончания поиска имеет вид .

**Математические модели в процедурах анализа на макроуровне**

[**Методы формирования математических моделей на макроуровне**](http://bigor.bmstu.ru/?met/?doc=110_Simul/2005.mod/?cou=140_CADedu/CAD.cou/?bck=110_Simul/2005.mod)

Исходную систему компонентных уравнений и топологических уравнений можно рассматривать как окончательную ММС, которая и подлежит численному решению. Численное решение этой системы уравнений предполагает *алгебраизацию* дифференциальных уравнений, например, с помощью преобразования Лапласа или формул численного интегрирования. В программах анализа нелинейных объектов на макроуровне, как правило, применяются формулы численного интегрирования, примером которых может служить неявная формула Эйлера:


где — значение переменной на -м шаге интегрирования; — шаг интегрирования. Алгебраизация подразумевает предварительную дискретизацию независимой переменной (вместо непрерывной переменной получаем конечное множество значений ), она заключается в представлении ММС в виде системы уравнений:


|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(20) |  (1) |


c неизвестными и , где использовано обозначение . Эту систему алгебраических уравнений, в общем случае нелинейных, необходимо решать на каждом шаге численного интегрирования исходных дифференциальных уравнений.

Однако порядок этой системы довольно высок и примерно равен , где — число ветвей эквивалентной схемы (каждая ветвь дает две неизвестные величины — фазовые переменные типа потока и типа потенциала, за исключением ветвей внешних источников, у каждой из которых неизвестна лишь одна фазовая переменная), — число элементов в векторе производных. Чтобы снизить порядок системы уравнений и тем самым повысить вычислительную эффективность ММС, желательно выполнить предварительное преобразование модели (в символическом виде) перед ее многошаговым численным решением. Предварительное преобразование сводится к исключению из системы части неизвестных и соответствующего числа уравнений. Оставшиеся неизвестные называют базисными. В зависимости от набора базисных неизвестных различают несколько методов формирования ММС.

Согласно *методу переменных состояния* (более полное название метода — метод переменных, характеризующих состояние), вектор базисных переменных состоит из *переменных состояния*. Этот вектор включает неизбыточное множество переменных, характеризующих накопленную в системе энергию. Например, такими переменными могут быть скорости тел (кинетическая энергия определяется скоростью, так как равна ), емкостные напряжения, индуктивные токи и т.п. Очевидно, что число уравнений не превышает . Кроме того, итоговая форма ММС оказывается приближенной к явной форме представления системы дифференциальных уравнений, т.е. к форме, в которой вектор явно выражен через вектор , что упрощает дальнейшее применение явных методов численного интегрирования. Метод реализуется путем особого выбора системы хорд и ветвей дерева при формировании топологических уравнений. Поскольку явные методы численного интегрирования дифференциальных уравнений не нашли широкого применения в программах анализа, то метод переменных состояния также теряет актуальность и его применение оказывается довольно редким.

В классическом варианте *узлового метода* в качестве базисных переменных используются узловые потенциалы (т.е. скорости тел относительно инерциальной системы отсчета, абсолютные температуры, перепады давления между моделируемой и внешней средой, электрические потенциалы относительно базового узла). Число узловых потенциалов и соответственно уравнений в ММС оказывается равным , где — число узлов в эквивалентной схеме. Обычно заметно меньше и, следовательно, порядок системы уравнений в ММС снижен более чем в два раза по сравнению с порядком исходной системы.

Однако классический вариант узлового метода имеет ограничения на применение и потому в современных программах анализа наибольшее распространение получил модифицированный узловой метод.

**Узловой метод**

Матрицу контуров и сечений в узловом методе формируют следующим образом. Выбирают базовый узел эквивалентной схемы и каждый из остальных узлов соединяют с базовым фиктивной ветвью. Именно фиктивные ветви принимают в качестве ветвей дерева, а все реальные ветви оказываются в числе хорд. Поскольку токи фиктивных ветвей равны нулю, а вектор напряжений фиктивных ветвей есть вектор узловых потенциалов , то топологические уравнения принимают вид:

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(35) |  (2) |

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(36) |  (3) |

где и — векторы напряжений и токов реальных ветвей.

Компонентные уравнения алгебраизуются с помощью одной из формул численного интегрирования, линеаризуются с помощью разложения в ряд Тейлора с сохранением только линейных членов, и их представляют в виде:

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(39) |  (4) |

где — диагональная матрица проводимостей ветвей, рассчитанная в точке ; — вектор, зависящий от значений фазовых переменных на предшествующих шагах интегрирования и потому уже известный к моменту времени . Каждая ветвь (за исключением идеальных источников напряжения) имеет проводимость, которая занимает одну из диагональных клеток матрицы проводимостей.

Окончательно ММС получаем, подставляя (4) и затем (2) в (3):

или

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(44) |  (5) |

где — матрица Якоби, — вектор правых частей. Отметим, что матрица имеет размер, матрица — , а матрица Якоби — .

Система (5) является системой линейных алгебраических уравнений, полученной в результате дискретизации независимой переменной, алгебраизации дифференциальных уравнений и линеаризации алгебраических уравнений. Алгебраизация приводит к необходимости пошагового вычислительного процесса интегрирования, линеаризация — к выполнению итерационного вычислительного процесса на каждом шаге интегрирования.

Рассмотрим, каким образом определяются проводимости ветвей.

Для резистивных ветвей проводимость — величина, обратная сопротивлению .

При использовании неявного метода Эйлера проводимость емкостной ветви получается из ее компонентного уравнения следующим образом.



На -м шаге интегрированияпроводимость по определению равна и при получаем . При этом в вектор правых частей входит элемент .

Проводимость индуктивной ветви можно найти аналогично:

и при 

Аналогично определяют проводимости и при использовании других разностных формул численного интегрирования, общий вид которых

где зависит от шага интегрирования, — от значений вектора на предыдущих шагах.

Классический вариант узлового метода имеет ограничения на применение. Так, недопустимы идеальные (с бесконечной проводимостью) источники напряжения, зависимые источники, аргументами которых являются токи, а также индуктивности, поскольку в классическом варианте токи не входят в число базисных переменных. Устранить эти ограничения довольно просто — нужно расширить совокупность базисных координат, включив в нее токи-аргументы зависимых источников, а также токи ветвей индуктивных и источников напряжения. Полученный вариант метода называют модифицированным узловым методом.

Согласно *модифицированному узловому методу*, в дерево при построении матрицы контуров и сечений включают ветви источников напряжения и затем фиктивные ветви. В результате матрица принимает вид (табл. 1), где введены обозначения: — источники напряжения, зависящие от тока; — независимые источники напряжения; — источники тока, зависящие от тока; — индуктивные ветви; — подматрица контуров хорд группы и сечений фиктивных ветвей группы .

Те же обозначения будем использовать и для соответствующих векторов напряжений и токов. Назовем ветви, токи которых являются аргументами в выражениях для зависимых источников, т.е. входят в вектор , особыми ветвями. Остальные ветви (за исключением индуктивных) — неособые. Введем также обозначения: — вектор индуктивных токов; и — векторы токов и напряжений неособых ветвей; — диагональные матрицы проводимостей ветвей неособых, индуктивных, особых.

**Таблица 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип ветви** | **Фиктивные** | **F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(75)** | **F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(76)** |
| **Неособые** | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(77) | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(78) | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(79) |
| **L** | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(80) | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(81) | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(82) |
| **F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(83)** | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(84) | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(85) | F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(86) |

Уравнение закона токов Кирхгофа (3) для фиктивных ветвей имеет вид


Исключим вектор с помощью компонентного уравнения (4), а вектор с помощью очевидного выражения:

где — матрица передаточных коэффициентов источников тока. Используем также выражение (2), принимающее вид

Получаем систему из трех матричных уравнений с неизвестными векторами , и :

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(92) |  (6) |

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(93) |  (7) |

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы формирования математических моделей на макроуровне_files\saved_resource(94) |  (8) |

где обозначено . Эта система и является итоговой ММ в узловом модифицированном методе.

**Примечание 1**

Вектор индуктивных токов нельзя исключить из итоговой системы уравнений, так как его значения входят в вектор на последующих шагах численного интегрирования.

**Примечание 2**

Источники тока, зависящие от напряжений, относятся к неособым ветвям, их проводимости входят в матрицу , которая при этом может иметь недиагональный вид.

**Примечание 3**

Источники напряжения, зависящие от напряжений, в приведенных выше выражениях не учитываются, при их наличии нужно в матрице выделить столбец для этих ветвей, что приводит к появлению дополнительных слагаемых в правых частях уравнений (6) — (8).

**Математическое обеспечения анализа на микроуровне**

[**Методы анализа на микроуровне**](http://bigor.bmstu.ru/?met/?doc=110_Simul/2013.mod/?cou=140_CADedu/CAD.cou/?bck=110_Simul/2013.mod)

В САПР решение дифференциальных или интегро-дифференциальных уравнений с частными производными выполняется численными методами. Эти методы основаны на дискретизации независимых переменных — их представлении конечным множеством значений в выбранных узловых точках исследуемого пространства. Эти точки рассматриваются как узлы некоторой сетки, поэтому используемые в САПР методы — это сеточные методы.

Среди сеточных методов наибольшее распространение получили два метода: метод конечных разностей (МКР) и метод конечных элементов (МКЭ). Обычно выполняют дискретизацию пространственных независимых переменных, т.е. используют пространственную сетку. В этом случае результатом дискретизации является система обыкновенных дифференциальных уравнений для задачи нестационарной или система алгебраических уравнений для стационарной.

Пусть необходимо решить уравнение  с заданными краевыми условиями  где и — дифференциальные операторы, — фазовая переменная, — вектор независимых переменных, () и () — заданные функции независимых переменных.

В *методе конечных разностей* алгебраизация производных по пространственным координатам базируется на аппроксимации производных конечно-разностными выражениями. При использовании метода нужно выбрать шаги сетки по каждой координате и вид шаблона. Под шаблоном понимают множество узловых точек, значения переменных в которых используются для аппроксимации производной в одной конкретной точке.

|  |
| --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы анализа на микроуровне_files\saved_resource(19) |

Рисунок 2. Примеры шаблонов для одномерных и двумерных задач

Примеры шаблонов для одномерных и двумерных задач приведены на рис. 2. На этом рисунке кружком большего диаметра обозначены узлы, в которых аппроксимируется производная. Черными точками обозначены узлы, значения фазовой переменной в которых входят в аппроксимирующее выражение. Число, записанное около узла, равно коэффициенту, с которым значение фазовой переменной входит в аппроксимирующее выражение. Так, для одномерных шаблонов в верхней части рисунка показана аппроксимация производной в точке , и указанным шаблонам при их просмотре слева направо соответствуют аппроксимации 

где — шаг дискретизации по оси .

Шаблоны для двумерных задач в нижней части рис. 1 соответствуют следующим конечно-разностным операторам:

* левый рисунок:



* средний рисунок:



* правый рисунок:



Здесь — значение в точке ; приняты одинаковые значения шагов по обеим координатам.

*Метод конечных элементов* основан на аппроксимации не производных, а самого решения . Но поскольку оно неизвестно, то аппроксимация выполняется выражениями с неопределенными коэффициентами 

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы анализа на микроуровне_files\saved_resource(34) |  (1) |

где — вектор-строка неопределенных коэффициентов, — вектор-столбец *координатных функций* (опорных функций), заданных так, что удовлетворяются граничные условия.

При этом речь идет об аппроксимациях решения в пределах конечных элементов, а с учетом их малых размеров можно говорить об использовании сравнительно простых аппроксимирующих выражений (например, — полиномы низких степеней). В результате подстановки в исходное дифференциальное уравнение и выполнения операций дифференцирования получаем систему невязок

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\3 Математическое обеспечение анализа проектных решений\Методы анализа на микроуровне_files\saved_resource(38) |  (2) |

из которой требуется найти вектор .

Эту задачу (определение ) решают одним из следующих методов:

* *метод коллокаций*, в котором, используя (2), формируют уравнений с неизвестным вектором :
где — число неопределенных коэффициентов;
* *метод наименьших квадратов*, основанный на минимизации квадратов невязок в точках или в среднем по рассматриваемой области;
* *метод Галеркина*, с помощью которого минимизируются в среднем по области невязки со специально задаваемыми весовыми коэффициентами.

Наибольшее распространение МКЭ получил в САПР машиностроения для анализа прочности объектов. Для этой задачи можно использовать рассмотренный подход, т.е. выполнить алгебраизацию исходного уравнения упругости (уравнения Ламе). Однако более удобным в реализации МКЭ оказался подход, основанный на вариационных принципах механики.

**Математическое обеспечения анализа на функционально-логическом уровне**

[**Классификация методов математического программирования**](http://bigor.bmstu.ru/?met/?doc=120_Opt/5003.mod/?cou=140_CADedu/CAD.cou/?bck=120_Opt/5003.mod)

Основными методами оптимизации в САПР являются поисковые методы. Поисковые методы основаны на пошаговом изменении управляемых параметров

где в большинстве методов приращение вектора управляемых параметров вычисляется по формуле

|  |  |
| --- | --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\5 Математическое обеспечение синтеза пректных решений\Классификация методов математического программирования_files\saved_resource(12) | (1) |

Здесь — значение вектора управляемых параметров на -м шаге; — шаг; а — направление поиска. Следовательно, если выполняются условия сходимости, то реализуется пошаговое (итерационное) приближение к экстремуму.

Методы оптимизации классифицируют по ряду признаков.

В зависимости от числа управляемых параметров различают методы одномерной и многомерной оптимизации. В методах *одномерной оптимизации* управляемый параметр единственный, во вторых размер вектора не менее двух. Реальные задачи в САПР многомерны, методы одномерной оптимизации играют вспомогательную роль на отдельных этапах многомерного поиска.

Различают методы условной и безусловной оптимизации по наличию или отсутствию ограничений. Для реальных задач характерно наличие ограничений, однако методы *безусловной оптимизации* также представляют интерес, поскольку задачи *условной оптимизации* с помощью специальных методов могут быть сведены к задачам без ограничений.

В зависимости от числа экстремумов различают задачи одно- и многоэкстремальные. Если метод ориентирован на определение какого-либо локального экстремума, то такой метод относится к локальным методам. Его называют методом *локального поиска*. Если же результатом является глобальный экстремум, то метод называют методом *глобального поиска*. Удовлетворительные по вычислительной эффективности *методы глобального поиска* для общего случая отсутствуют и потому на практике в САПР используют методы поиска локальных экстремумов.

Наконец, в зависимости от того, используются при поиске производные целевой функции по управляемым параметрам или нет, различают методы нескольких порядков. Если производные не используются, то имеет место метод нулевого порядка, если используются первые или вторые производные, то соответственно метод первого или второго порядка. Методы первого порядка называют также градиентными, поскольку вектор первых производных по есть градиент целевой функции


Конкретные методы определяются следующими факторами:

1. способом вычисления направления поиска в формуле (1);
2. способом выбора шага ;
3. способом определения окончания поиска.

Определяющим фактором является первый из перечисленных в этом списке.

Шаг может быть или постоянным, или выбираться исходя из одномерной оптимизации — поиска минимума целевой функции в выбранном направлении . В последнем случае шаг будем называть оптимальным.

Окончание поиска обычно осуществляют по правилу: если на протяжении подряд идущих шагов траектория поиска остается в малой -окрестности текущей точки поиска , то поиск следует прекратить, следовательно,. условие окончания поиска имеет вид .

**Тема 1.4. Математическое обеспечение синтеза проектных решений**

**Методы оптимизации**

[**Требования к математическим моделям и методам в САПР**](http://bigor.bmstu.ru/?met/?doc=190_CAD/2001.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/2001.mod)

Основными требованиями к математическим моделям являются требования адекватности, точности, экономичности.

Модель всегда лишь приближенно отражает некоторые свойства объекта. Адекватность имеет место, если модель отражает заданные свойства объекта с приемлемой точностью. Под точностью понимают степень соответствия оценок одноименных свойств объекта и модели.

Экономичность (*вычислительная эффективность*) определяется затратами ресурсов, требуемых для реализации модели. Поскольку в САПР используются математические модели, далее речь пойдет о характеристиках именно математических моделей, и экономичность будет характеризоваться затратами машинных времени и памяти.

Адекватность оценивается перечнем отражаемых свойств и *областями адекватности*. Область адекватности — область в пространстве параметров, в пределах которой погрешности модели остаются в допустимых пределах. Например, область адекватности линеаризованной модели поверхности детали определяется системой неравенств:

где , и — -я координата -й точки поверхности в объекте и модели соответственно; и — допущенная и предельно допустимая относительные погрешности моделирования поверхности, максимум берется по всем координатам и контролируемым точкам.

Отметим, что в большинстве случаев области адекватности строятся в пространстве внешних переменных. Так, область адекватности модели электронного радиоэлемента обычно выражает допустимые для применения модели диапазоны изменения моделируемых температур, внешних напряжений, частот.

Аналогичные требования по точности и экономичности фигурируют при выборе численных методов решения уравнений модели.

**Постановка задач параметрического и структурного синтеза**

[**Исчисления**](http://bigor.bmstu.ru/?met/?doc=190_CAD/6005.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/6005.mod)

Очевидно, что в большинстве случаев структурного синтеза вместо нереализуемого явного представления всего множества проектных решений задают множество элементов и совокупность правил объединения этих элементов в допустимые структуры (проектные решения).

Эти множества элементов и правил часто представляют в виде *формальной системы* (исчисления), т.е. задача синтеза имеет вид

где — алфавит исчисления (алфавит представлен базовыми элементами, из которых синтезируется структура); — подмножество выражений, называемых формулами исчисления и получаемых на основе синтаксических правил (или множество слов, не совпадающих с буквами алфавита и служащих для обозначения переменных); — множество аксиом исчисления, под которыми понимаются задаваемые исходные формулы (слова) в алфавите (например, соответствия функций и элементов); — множество правил вывода новых формул в алфавите из аксиом и ранее выведенных корректных формул. Каждую формулу можно интерпретировать как некоторую структуру, поэтому синтез — это процесс вывода формулы, удовлетворяющей исходным требованиям и ограничениям.

Другие примеры компактного задания множества альтернатив через множества и связаны с использованием систем искусственного интеллекта, в которых есть база данных, — база знаний, или эволюционных методов, в которых — также база данных, — множество эвристик, последовательность применения которых определяется эволюционными и генетическими принципами.

[**Эволюционные методы**](http://bigor.bmstu.ru/?met/?doc=045_ga/1010.mod/?cou=140_CADedu/CAD.cou/?bck=045_ga/1010.mod)

*Эволюционные методы* (ЭМ) являются приближенными (эвристическими) методами решения задач оптимизации и структурного синтеза. Большинство ЭМ основано на статистическом подходе к исследованию ситуаций и итерационном приближении к искомому решению.

Эволюционные вычисления составляют один из разделов искусственного интеллекта. При построении систем ИИ по данному подходу основное внимание уделяется построению начальной модели, и правилам, по которым она может изменяться (эволюционировать). Причем модель может быть составлена по самым различным методам, например, это может быть и нейронная сеть и набор логических правил. К основным эволюционным методам относятся методы отжига, генетические, поведения "толпы" (PSO), колонии муравьев (ACO), генетического программирования.

В отличие от точных методов математического программирования ЭМ позволяют находить решения, близкие к оптимальным, за приемлемое время, а в отличие от других эвристических методов оптимизации характеризуются существенно меньшей зависимостью от особенностей приложения (т.е. более универсальны) и в большинстве случаев обеспечивают лучшую степень приближения к оптимальному решению. Универсальность ЭМ определяется также применимостью к задачам с неметризуемым пространством управляемых переменных (т.е. среди управляемых переменных могут быть и лингвистические величины, т.е. не имеющие количественного выражения).

В *методе отжига* (Simulated Annealing) имитируется процесс минимизации потенциальной энергии тела во время отжига деталей. В текущей точке поиска происходит изменение некоторых управляемых параметров. Новая точка принимается всегда при улучшении целевой функции и лишь с некоторой вероятностью при ее ухудшении.

Важнейшим частным случаем ЭМ являются *генетические методы* и алгоритмы. Генетические алгоритмы (ГА) основаны на поиске лучших решений с помощью наследования и усиления полезных свойств множества объектов определенного приложения в процессе имитации их эволюции.

Свойства объектов представлены значениями параметров, объединяемых в запись, называемую в ЭМ *хромосомой*. В ГА оперируют подмножеством хромосом, называемом популяцией. Имитация генетических принципов — вероятностный выбор родителей среди членов популяции, скрещивание их хромосом, отбор потомков для включения в новые поколения объектов на основе оценки целевой функции — ведет к эволюционному улучшению значений целевой функции (функции полезности) от поколения к поколению.

Среди ЭМ находят применение также методы, которые в отличие от ГА оперируют не множеством хромосом, а единственной хромосомой. Так, метод дискретного локального поиска (его англоязычное название Hillclimbing) основан на случайном изменении отдельных параметров (т.е. значений полей в записи или, другими словами, значений генов в хромосоме). Такие изменения называют мутациями. После очередной мутации оценивают значение функции полезности (Fitness Function) и результат мутации сохраняется в хромосоме только, если улучшилась. При "моделировании отжига" результат мутации сохраняется с некоторой вероятностью, зависящей от полученного значения .

В методе PSO (Particles Swarm Optimization) имитируется поведение множества агентов, стремящихся согласовать свое состояние с состоянием наилучшего агента.

Метод колонии муравьев (ACO) основан на имитации поведения муравьев, минимизирующих длину своих маршрутов на пути от муравьиной кучи до источника пищи.

**Тема 1.5. Системные среды САПР**

[**Требования к техническому обеспечению САПР**](http://bigor.bmstu.ru/?met/?doc=190_CAD/1001.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/1001.mod)

Техническое обеспечение САПР включает в себя различные технические средства (hardware), используемые для выполнения автоматизированного проектирования, а именно вычислительные системы, ЭВМ (компьютеры), периферийные устройства, сетевое оборудование, а также оборудование некоторых вспомогательных систем (например, измерительных), поддерживающих проектирование. Отметим, что *вычислительной системой* (в отличие от ЭВМ и вычислительной сети) называют совокупность аппаратных и программных средств, совместно используемых при решении задач и размещаемых компактно на территории, размеры которой соизмеримы с размерами аппаратных средств.

Используемые в САПР технические средства должны обеспечивать:

* выполнение всех необходимых проектных процедур, для которых имеется соответствующее ПО;
* взаимодействие между проектировщиками и ЭВМ, поддержку интерактивного режима работы;
* взаимодействие между членами коллектива, выполняющими работу над общим проектом.

Первое из этих требований выполняется при наличии в САПР вычислительных машин и систем с достаточными производительностью и емкостью памяти.

Второе требование относится к пользовательскому интерфейсу и выполняется за счет включения в САПР удобных средств ввода-вывода данных и прежде всего устройств обмена графической информацией.

Третье требование обусловливает объединение аппаратных средств САПР в вычислительную сеть.

В результате общая структура ТО САПР представляет собой сеть узлов, связанных между собой средой передачи данных. Узлами (станциями данных) являются рабочие места проектировщиков, часто называемые *автоматизированными рабочими местами* (АРМ) или рабочими станциями (WS — Workstation), ими могут быть также большие ЭВМ (мейнфреймы), отдельные периферийные и измерительные устройства. Именно в АРМ должны быть средства для интерфейса проектировщика с ЭВМ. Что касается вычислительной мощности, то она может быть распределена между различными узлами вычислительной сети.

Среда передачи данных представлена каналами передачи данных, состоящими из линий связи и коммутационного оборудования.

**Назначение и состав системных сред САПР**

[**Структура САПР**](http://bigor.bmstu.ru/?met/?doc=190_CAD/0009.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/0009.mod)

Как и любая сложная система, САПР состоит из подсистем. Различают подсистемы проектирующие и обслуживающие.

*Проектирующие подсистемы* непосредственно выполняют проектные процедуры. Примерами проектирующих подсистем могут служить подсистемы геометрического трехмерного моделирования механических объектов, изготовления конструкторской документации, схемотехнического анализа, трассировки соединений в печатных платах.

*Обслуживающие подсистемы* обеспечивают функционирование проектирующих подсистем, их совокупность часто называют системной средой (или оболочкой) САПР. Типичными обслуживающими подсистемами являются подсистемы управления проектными данными, подсистемы разработки и сопровождения программного обеспечения CASE (Computer Aided Software Engineering), обучающие подсистемы для освоения пользователями технологий, реализованных в САПР.

Структурирование САПР по различным аспектам обусловливает появление *видов обеспечения САПР*. Принято выделять семь видов обеспечения:

* *техническое обеспечение* (ТО), включающее различные аппаратные средства (ЭВМ, периферийные устройства, сетевое коммутационное оборудование, линии связи, измерительные средства);
* *математическое обеспечение* (МО), объединяющее математические методы, модели и алгоритмы для выполнения проектирования;
* *программное обеспечение* (ПО), представляемое компьютерными программами САПР;
* *информационное обеспечение* (ИО), состоящее из баз данных (БД), систем управления базами данных (СУБД), а также включающее другие данные, используемые при проектировании;
* *лингвистическое обеспечение* (ЛО), выражаемое языками общения между проектировщиками и ЭВМ, языками программирования и языками обмена данными между техническими средствами САПР;
* *методическое обеспечение* (МетО), включающее различные методики проектирования, иногда к МетО относят также математическое обеспечение;
* *организационное обеспечение* (ОО), представляемое штатными расписаниями, должностными инструкциями и другими документами, регламентирующими работу проектного предприятия.

Отметим, что вся совокупность используемых при проектировании данных называется информационным фондом САПР. Базой данных называют упорядоченную совокупность данных, отображающих свойства объектов и их взаимосвязи в некоторой предметной области. Доступ к БД для чтения, записи и модификации данных осуществляется с помощью СУБД, а совокупность БД и СУБД называют банком данных (БнД).

Классификацию САПР осуществляют по ряду признаков, например, по приложению, целевому назначению, масштабам (комплексности решаемых задач), характеру базовой подсистемы — ядра САПР.

По приложениям наиболее представительными и широко используемыми являются следующие группы САПР.

1. САПР для применения в отраслях общего машиностроения. Их часто называют *машиностроительными САПР* или MCAD (Mechanical CAD) системами.
2. *САПР в области радиоэлектроники*: системы ECAD (Electronic CAD) или EDA (Electronic Design Automation).
3. САПР в области архитектуры и строительства.

Кроме того, известно большое число специализированных САПР, или выделяемых в указанных группах, или представляющих самостоятельную ветвь в классификации. Примерами таких систем являются САПР больших интегральных схем (БИС); САПР летательных аппаратов; САПР электрических машин и т.п.

По целевому назначению различают САПР или подсистемы САПР, обеспечивающие разные аспекты (страты) проектирования. Так, в составе MCAD появляются CAE/CAD/CAM системы.

По масштабам различают отдельные *программно-методические комплексы* (ПМК) САПР, например, комплекс анализа прочности механических изделий в соответствии с методом конечных элементов (МКЭ) или комплекс анализа электронных схем; системы ПМК; системы с уникальными архитектурами не только программного (software), но и технического (hardware) обеспечений.

По характеру базовой подсистемы различают следующие разновидности САПР.

1. САПР на базе подсистемы машинной графики и геометрического моделирования. Эти САПР ориентированы на приложения, где основной процедурой проектирования является конструирование, т.е. определение пространственных форм и взаимного расположения объектов. Поэтому к этой группе систем относится большинство САПР в области машиностроения, построенных на базе графических ядер. В настоящее время широко используются унифицированные графические ядра, применяемые более чем в одной САПР, это ядра Parasolid фирмы EDS Unigraphics и ACIS фирмы Intergraph.
2. САПР на базе СУБД. Они ориентированы на приложения, в которых при сравнительно несложных математических расчетах перерабатывается большой объем данных. Такие САПР преимущественно встречаются в технико-экономических приложениях, например, при проектировании бизнес-планов, но имеют место также при проектировании объектов, подобных щитам управления в системах автоматики.
3. САПР на базе конкретного прикладного пакета. Фактически это автономно используемые программно-методические комплексы, например, имитационного моделирования производственных процессов, расчета прочности по методу конечных элементов, синтеза и анализа систем автоматического управления и т.п. Часто такие САПР относят к системам CAE. Примерами могут служить программы логического проектирования на базе языка VHDL, математические пакеты типа MathCAD.
4. Комплексные (интегрированные) САПР, состоящие из совокупности подсистем предыдущих видов. Характерными примерами комплексных САПР являются CAE/CAD/CAM-системы в машиностроении или САПР БИС. Так, САПР БИС включает в себя СУБД и подсистемы проектирования компонентов, принципиальных, логических и функциональных схем, топологии кристаллов, тестов для проверки годности изделий. Для управления столь сложными системами применяют специализированные системные среды.

[**Логистические системы**](http://bigor.bmstu.ru/?met/?doc=020_ERP/erp105.mod/?cou=140_CADedu/CAD.cou/?bck=020_ERP/erp105.mod)

Сложность задач управления, которые приходится решать в современных производственных системах, обусловливает интерес к логистике и развитию логистических систем. Традиционно *логистику* связывали с управлением процедурами движения сырья от источников снабжения к месту производства продукции, ее складирования и движения от производственной линии к месту потребления. При этом логистику определяют как науку об организации менеджмента для эффективного продвижения продукции по цепи "закупки сырья — производство продукции – сбыт — распределение" с целью минимизации общих затрат ресурсов и удовлетворения требований конечных потребителей к качеству продукции и услуг. Функции и задачи, выполняемые в подсистемах "Логистика" большинства систем ERP, соответствуют этому определению.

Однако часто используется и более широкая трактовка логистики, как науки об управлении и оптимизации материальных потоков, потоков услуг и связанных с ними финансовых и информационных потоков при функционировании различных систем. Системы, реализующие управление названными потоками, называют *логистическими системами*. Основными логистическими функциями являются поддержка послепродажного обслуживания потребителей продукции, управление закупками и заказами, транспортировка, управление запасами, управление производственными процедурами, ценообразование, распределение готовой продукции в товаропроводящих структурах предприятия и посредников. При этом под управлением производственными процедурами подразумевается решение задач объемно-календарного планирования, минимизации уровня запасов материальных ресурсов и незавершенного производства, расчета потребностей в материалах и т.п.

В зависимости от того, на каких процедурах и каких потоках акцентируется внимание, различают ряд разделов логистики. Так, используют понятия логистик внутренней, внешней, производственной, транспортной и т.п. Для автоматизированных систем проектирования и управления, оперирующих информационными ресурсами, наиболее актуальны задачи информационной логистики. Под *информационной логистикой* понимают организацию и использование систем информационного обеспечения производственно-хозяйственных процессов на предприятии. Другими словами, в информационной логистике акцент делается на информационные потоки, т.е. на потоки документов (в различных формах), порождаемых материальными потоками и сопровождающих материальные потоки. Информационная логистика базируется на системном подходе, который охватывает все виды деятельности, связанные с планированием и управлением процессами, нацеленными на обеспечение предприятия релевантной информацией.

Во многих приложениях управление данными в процессах проектирования, производства и эксплуатации сложных изделий ведется с первоочередной целью минимизации издержек, имеющих место на этапе эксплуатации. Собственно минимизация может быть обеспечена должным учетом логистических требований на этапах проектирования и изготовления изделий.

Примерами логистических систем являются системы *MRP* (часто обозначаемая MRP-1) и *MRP-2*. В системе MRP-1 исходными данными являются сведения о заказах потребителей, объем запасов, объем и сроки производства конечной продукции, а результатом работы системы — необходимое количество материалов и компонентов (точнее, заказ на их изготовление или приобретение). Система MRP-2 — логистическая система, объединяющая функции MRP-1, планирование производственных ресурсов и финансов, контроль над всеми стадиями производственного процесса с целью минимизации издержек.

Логистическая система MRP-2 — система для дискретных производств различных типов. Она может быть самостоятельной системой или составной частью системы ERP. Основные функции (подсистемы) MRP-2:

* управление спросом (прогнозирование, работа с заказами);
* планирование продаж и операций;
* календарное планирование (расчет календарных планов основан на синтезе расписаний, при диспетчировании и запуске заказов на рабочих центрах устанавливаются приоритеты заказов, примеры правил — FIFO, SPT — Shortest Processing Time — минимальное время обработки заказа, STPT — Shortest Total Processing Time Remaining — минимальное оставшееся время обработки, EDD — Earliest Due Time — минимальный срок исполнения заказа и др.);
* планирование потребностей в материалах, формирование BOM;
* планирование потребностей в мощностях (каждой позиции календарного плана сопоставляется список ресурсов с указанием требуемого объема и сроков использования);
* управление запасами (запасы характеризуются местом хранения и объемом, их сопровождение в MRP-2 включает операции: отпуск со складов по наряду, на продажу, возврат поставщику, получение по заказу из производства, заказу на закупки, возврат от покупателя, списание, корректировка из-за потерь, изменения цен и т.п.)
* управление снабжением (типичный процесс снабжения включает шаги: выявление потребности, определение потенциальных поставщиков, получение от них коммерческих предложений. выбор поставщика, формирование договора, контроль исполнения заказа);
* инструментальное обеспечение;
* интерфейс с финансовым планированием;
* оперативное управление производством;
* моделирование процессов;
* оценка деятельности предприятия.

**Раздел 2. Инструментарий проектировщика**

**Тема 2.1. Методики проектирования автоматизированных систем**

[**Типы САПР в области машиностроения**](http://bigor.bmstu.ru/?met/?doc=190_CAD/8004.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/8004.mod)

Среди CAD-систем в машиностроении (MCAD) различают системы нижнего, среднего и верхнего уровней. Это разделение возникло на рубеже 80-90-х годов прошлого века. Системами нижнего уровня (или легкими системами) стали называть сравнительно дешевые САПР, ориентированные на 2D-графику, т.е. на автоматизацию преимущественно чертежных работ. Техническим обеспечением легких САПР были персональные ЭВМ, в то время значительно уступавшие по своим возможностям рабочим станциям.

Системы верхнего уровня, называемые также *"тяжелыми" САПР* (или high-end), разрабатывались для реализации на рабочих станциях или мейнфреймах. Эти системы были более универсальными, но и дорогими, ориентированными на геометрическое твердотельное и поверхностное моделирование. Оформление чертежной документации в них обычно осуществляется с помощью предварительной разработки трехмерных геометрических моделей. В дальнейшем системы, в которых 3D-моделирование ограничивалось лишь твердотельными моделями, т.е. занимавшие промежуточное положение между "легкими" и "тяжелыми" САПР, стали называть системами среднего уровня.

В настоящее время развитие САПР привело к тому, что во многих системах среднего уровня появились средства поверхностного моделирования, а возможности персональных ЭВМ стали приемлемыми для систем верхнего уровня. В результате изменились принципы, по которым различают тяжелые и средние системы. Тяжелыми теперь называют системы CAE/CAD/CAM/PDM, т.е. системы с возможностями конструкторского и технологического проектирования, инженерного анализа, управления проектными данными и с расширенным составом специализированных программных модулей в подсистемах CAD и CAM. В отличие от них, системы среднего уровня теперь называют также серийными, mainstream или mid-range.

К классу high-end систем сегодня CATIA, Unigraphics NX, ProEngineer, а к mainstream системам - SolidWorks, SolidEdge, Inventor и ряд других.

Системы одного уровня по своим функциональным возможностям приблизительно равноценны, новые достижения, появившиеся в одном из программно-методических комплексов САПР, в скором времени реализуются в новых версиях других комплексов.

В САПР крупных предприятий обычно используют программы разных уровней. Связано это с тем, что более 80% всех процедур конструирования можно выполнить на CAD-системах нижнего и среднего уровней, кроме того, "тяжелые" системы дороги. Поэтому предприятие приобретает лишь ограниченное число экземпляров (лицензий) программы верхнего уровня, а большинство клиентских рабочих мест обеспечивается экземплярами программ нижнего или среднего уровней. При этом возникает проблема обмена информацией между разнотипными CAD-системами. Она решается путем использования языков и форматов, принятых в CALS-технологиях, хотя для неискаженной передачи геометрических данных с помощью промежуточных унифицированных языков приходится преодолевать определенные трудности.

**Средства системного этапа проектирования**

[**Спецификации проектов программных систем**](http://bigor.bmstu.ru/?met/?doc=190_CAD/7005.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/7005.mod)

Важное значение в процессе разработки ПО имеют средства спецификации проектов ПО. Средства спецификации в значительной мере определяют суть методов CASE.

Способы и средства спецификации классифицируют по базовой методологии, используемой для декомпозиции ПО, как сложной системы, и по аспектам моделирования ПО.

Различают два подхода к декомпозиции ПО. Первый способ называют функциональным или структурным. Он основан на выделении функций и потоков данных. Второй способ — объектный, выражает идеи объектно-ориентированного проектирования и программирования.

Аспектами моделирования приложений являются функциональное, поведенческое и информационное описания.

Практически все способы функциональных спецификаций имеют следующие общие черты:

* модель имеет иерархическую структуру, представляемую в виде диаграмм нескольких уровней;
* элементарной частью диаграммы каждого уровня является конструкция вход-функция-выход;
* необходимая дополнительная информация содержится в файлах поясняющего текста.

В большинстве случаев функциональные диаграммы являются *диаграммами потоков данных* (DFD — Data Flow Diagram). В DFD блоки (прямоугольники) соответствуют функциям, дуги — входным и выходным потокам данных. Поясняющий текст представлен в виде "словарей данных", в которых указаны компонентный состав потоков данных, число повторений циклов и т.п. Для описания структуры информационных потоков можно использовать нотацию Бэкуса-Наура.

Одна из нотаций для DFD предложена Е.Йорданом. В ней описывают процессы (функции), потоки данных, хранилища и внешние сущности, их условные обозначения показаны на рис. 3.

|  |
| --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\7 Методическое и программное обеспечение\Спецификации проектов программных систем_files\saved_resource(10) |

Рисунок 3.  Изображения элементов в нотации Йордана

Разработка DFD начинается с построения диаграммы верхнего уровня, отражающей связи программной системы, представленной в виде единого процесса, с внешней средой. Декомпозиция процесса проводится до уровня, на котором фигурируют элементарные процессы, которые могут быть представлены одностраничными описаниями алгоритмов (миниспецификациями) на терминальном языке программирования.

Для описания информационных моделей наибольшее распространение получили диаграммы сущность-отношение (ERD — Entity-Relation Diagrams), в которых предусмотрены средства для описания сущностей, атрибутов и отношений. Спецификации хранилищ данных в CASE, как правило, даются с помощью диаграмм сущность-отношение. Стандартной методикой построения таких диаграмм является IDEF1X.

Поведенческие модели описывают процессы обработки информации. В инструментальных CASE-системах их представляют в виде граф-схем, диаграмм перехода состояний, таблиц решений, псевдокодов (языков спецификаций), процедурных языков программирования, в том числе языков четвертого поколения.

В граф-схемах блоки, как и в диаграммах DFD, используют для задания процессов обработки, но дуги имеют иной смысл — они описывают последовательность передач управления (вместе с специальными блоками управления).

В *диаграммах перехода состояний* узлы соответствуют состояниям моделируемой системы, дуги — переходам из состояние в состояние, атрибуты дуг — условиям перехода и инициируемым при их выполнении действиям. Очевидно, что как и в других конечно-автоматных моделях, кроме графической формы представления диаграмм перехода состояний, можно использовать также табличные формы. Так, при изоморфном представлении с помощью таблиц перехода состояний каждому переходу соответствует строка таблицы, в которой указываются исходное состояние, условие перехода, инициируемое при этом действие и новое состояние после перехода.

Близкий по своему характеру способ описания процессов основан на *таблицах решений* (или деревьях решений). Каждый столбец таблицы решений соответствует определенному сочетанию условий, при выполнении которых осуществляются действия, указанные в нижерасположенных клетках столбца.

Таблицы решений удобны при описании процессов с многократными ветвлениями. В этих случаях помогают также визуальные языки программирования, в которых для описания процессов используют графические элементы, подобные приведенным на рис. 4.

|  |
| --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\7 Методическое и программное обеспечение\Спецификации проектов программных систем_files\saved_resource(11) |

Рисунок 4.  Примеры описания операторов в визуальных языках программирования

В псевдокодах алгоритмы записываются с помощью как средств некоторого языка программирования (преимущественно для управляющих операторов), так и естественного языка (для выражения содержания вычислительных блоков). Используются конструкции (операторы) следования, условные, цикла. Служебные слова из базового языка программирования или из DFD записываются заглавными буквами, фразы естественного языка — строчными.

Языки четвертого поколения предназначены для описания программ как совокупностей заранее разработанных программных модулей. Поэтому одна команда языка четвертого поколения может соответствовать значительному фрагменту программы на языке 3GL. Примерами языков 4GL могут служить Informix-4GL, JAM, NewEra, XAL.

Миниспецификации процессов могут быть выражены с помощью псевдокодов (языков спецификаций), визуальных языков проектирования или языков программирования,

Объектный подход представлен компонентно-ориентированными технологиями разработки ПО. При объектном подходе ПО формируется из компонентов, объединяющих в себе алгоритмы и данные и взаимодействующих путем обмена сообщениями. Для поддержки объектного подхода разработан стандартный язык моделирования приложений UML.

**Инструментальные средства концептуального проектирования**

[**CRM — системы взаимоотношений с заказчиками**](http://bigor.bmstu.ru/?met/?doc=020_ERP/erp107.mod/?cou=140_CADedu/CAD.cou/?bck=020_ERP/erp107.mod)

Система *CRM* — это система, на вход которой поступают данные, связанные с клиентами компании, а на выходе появляется информация, влияющая на поведение компании в целом или на поведение ее отдельных элементов (вплоть до конкретного работника компании). Другими словами, CRM-система — это прежде всего база данных с информацией о клиентах. и набор приложений, которые позволяют, во-первых, собирать информацию о клиенте, во-вторых, ее обрабатывать, в третьих, делать определенные выводы на базе этой информации, экспортировать ее в другие приложения или просто при необходимости предоставлять эту информацию в удобном виде. Собственно, эти моменты и являются ключевыми функциями CRM-систем. Результатами работы CRM-системы могут пользоваться не только сотрудники компании, но и непосредственно сам клиент. Причем современные технологии позволяют это сделать без какой-либо помощи сотрудников организации.

Существуют ERP-системы в которых CRM выполняют роль подсистем (например, R3, Oracle E-Business Suite, Axapta) однако в настоящее время чаще CRM являются самостоятельными системами (например, Siebel, WinPeak CRM, Sales Expert).

Можно выделить три типа стандартных CRM-систем:

1. Оперативное использование. Система используется сотрудником компании для оперативного доступа к информации по конкретному клиенту в ходе непосредственного взаимоотношения с клиентом — процессов продажи и обслуживания.

Информация по клиенту включает, как минимум, такие составляющие, как:

* название компании;
* рубрицированные свойства компании (отрасль, регион и т.д.);
* контактные лица;
* хронологическая история взаимоотношений с этой компанией.

Основным компонентом системы является приложение, которое в наглядном виде предоставляет сотруднику указанную информацию. Для удобства работы с информацией организуется много специальных возможностей:

* различные виды поиска, сортировки и экспорт данных о компаниях и персонах;
* групповые и индивидуальные планировщики, будильники, календари;
* конструкторы отчетов для решения управленческих и маркетинговых задач.

Дополнительно СRM-система может быть использована для ввода, хранения и анализа информации об отгрузках товаров, оказании услуг, поступлении денег, выполнении заказов и проектов и других задач. Эта информация может вводиться вручную или импортироваться из других информационных систем автоматически.

В первую очередь, от системы требуется хорошая интеграция между всеми подсистемами и возможность пополнять базу данных в процессе любого взаимодействия с клиентом. Данный тип CRM-систем является наиболее распространенным в традиционном бизнесе.

2. Аналитическое использование. Система используется для анализа различных данных, относящихся как к самому клиенту/клиентам, так и к деятельности фирмы. Осуществляется поиск статистических закономерностей в этих данных для выработки наиболее эффективной стратегии маркетинга, продаж, обслуживания клиентов и т. д. Требуется хорошая интеграция подсистем, большой объем наработанных статистических данных, эффективный аналитический инструментарий, интеграция с другими системами, автоматизирующими деятельность предприятия. Данные, генерируемые такими системами, могут быть затребованы отделом маркетинга, представлены клиенту без посредничества сотрудников компании. Такой тип CRM-систем чаще применяется в электронной коммерции, нежели в традиционном бизнесе

3. Коллаборационное использование. CRM-системы предоставляют клиентам возможность гораздо большего влияния на деятельность фирмы в целом, в том числе на процессы разработки дизайна, производства, доставки и обслуживания продукта. Для этого требуются технологии, позволяющие с минимальными затратами подключить клиента к сотрудничеству в рамках внутренних процессов компании. Клиент нередко использует сеть Интернет для доступа к таким системам, и они наиболее распространены в сфере электронной коммерции.

Например, с помощью Web-витрин в составе систем eCRM клиенту предоставляется возможность просмотра товаров фирмы. Основной функцией eCRM-системы является правильное расположение товаров и обеспечение удобной навигации по Web-сайту. Интернет-технологии позволяют определить клиента (его компьютер), зашедшего на сайт магазина, и, соответственно, кастомизировать представление товара на витрине, интерфейс сайта и пр. На главной странице сайта можно приветствовать клиента по имени, автоматически определять регион покупателя и доступные в этом регионе товары, извещать клиента о наиболее интересных для него новинках магазина.

**Маршрут проектирования ПЛИС и возможности типовых САПР**

[**Конструкторское проектирование СБИС**](http://bigor.bmstu.ru/?met/?doc=010_EDA/eda042.mod/?cou=140_CADedu/CAD.cou/?bck=010_EDA/eda042.mod)

Основой программного обеспечения конструкторского проектирования в системах ECAD являются средства топологического проектирования, среди которых выделяют программы разработки топологии (layout) кристаллов СБИС, многокристальных СБИС и печатных плат.

Конструкторское проектирование СБИС включает в себя ряд процедур. *Разрезание* (partitioning или компоновка) заключается в группировании компонентов по критерию связности, что необходимо или для *размещения* формируемых групп в отдельных чипах при многокристальной реализации, или для определения их взаимного расположения в одном кристалле в процессе выполнения последующей процедуры планировки кристалла (floorplanning). Группы при планировании представляют в виде прямоугольников, их расположение обычно определяется в интерактивном режиме, но находят применение также генетические алгоритмы.

Далее следуют процедуры размещения (placement) компонентов, трассировки (routing) соединений, сжатия (compaction) топологической схемы, проверки соответствия топологической и принципиальной схем, подготовки информации для генераторов изображений. Ответственность процедуры размещения определяется доминирующим влиянием на быстродействие СБИС задержек именно в соединениях. *Трассировка* состоит из глобальной фазы, во время которой намечается положение трасс, и детальной, которая, в свою очередь, подразделяется на канальную (channel) и локальную (switchbox). Канальная трассировка служит для конкретизации положения трасс в каналах, а локальная — для проведения соединений между каналами и контактами компонентов. Сжатие топологии выполняется во всех направлениях и позволяет уменьшить занимаемую схемой площадь. После исполнения операций размещения и трассировки следует оценка задержек и занимаемой площади и, если требования к этим параметрам не удовлетворены, то дополнительно увеличивают число итераций для приближения к оптимальным результатам.

При конструкторском проектировании для описания топологии СБИС широкую известность получили *форматы EDIF* (Electronic Design Interchange Format) и CIF (Caltech Intermediate Format). Формат EDIF удобен для передачи данных, включающих списки соединений, параметры СБИС или печатных плат, спецификации тестовых наборов, результаты моделирования и т.п. Формат CIF применяют при передаче проекта, представленного на уровне геометрических примитивов и управляющих данных, в производство.

Моделирование технологических процессов изготовления СБИС относят к технологическому проектированию, поддерживаемому соответствующими программами ECAD.

**Тема 2.2. Основные сведения о языке VHDL**

Языки описания аппаратуры (Hardware Description Language), является формальной записью, которая может быть использована на всех этапах разработки цифровых электронных систем. Это возможно вследствие того, что язык легко воспринимается как машиной, так и человеком он может использоваться на этапах проектирования, верификации, синтеза и тестирования аппаратуры также как и для передачи данных о проекте, модификации и сопровождения. Наиболее универсальным и распространенным языком описания аппаратуры является VHDL. На этом языке возможно как поведенческое, так структурное и потоковое описание цифровых схем.

Язык VHDL используется во многих системах для моделирования цифровых схем, проектирования программируемых логических интегральных микросхем, базовых матричных кристаллов, заказных интегральных микросхем

С точки зрения программиста язык VHDL состоит как бы из двух компонент – общеалгоритмической и проблемно-ориентированной.

Общеалгоритмическая компонента VHDL- это язык, близкий по синтаксису и семантике к современным языкам программирования типа Паскаль, C и др. Язык относится к классу строго типизированных. Помимо встроенных (пакет STANDART) простых (скалярных) типов данных: целый, вещественный, булевский, битовый, данных типа время, данных типа ссылка (указатель) пользователь может вводить свои типы данных (перечислимый, диапазонный и др.).

Помимо скалярных данных можно использовать агрегаты: массивы array, в том числе и битовые векторы bit\_vector, и символьные строки string, записи record, файлы file.

Последовательно выполняемые (последовательные) операторы VHDL могут использоваться в описании процессов, процедур и функций. Их состав включает:

* оператор присваивания переменной (:=);
* последовательный оператор назначения сигналу (<=);
* последовательный оператор утверждения (assert);
* условный (if);
* выбора (case);
* цикла (loop);
* пустой оператор (null);
* оператор возврата процедуры- функции (return);
* оператор последовательного вызова процедуры.

Язык поддерживает концепции пакетного и структурного программирования. Сложные операторы заключены в операторные скобки: if- end if; process- end process; case- end case; loop- end loop и т. д.

Различаются локальные и глобальные переменные. Область “видимости” локальных переменных ограничена пределами блока (процессного, процедурного, оператора блока, оператора описания архитектуры).

Фрагменты описаний, которые могут независимо анализироваться компилятором и при отсутствии ошибок помещаться в библиотеку проекта (рабочую библиотеку Work),называются проектными пакетами design unit. Такими пакетами могут быть объявление интерфейса объекта проекта entity, объявление архитектуры architecture, объявление конфигурации configuration, объявление интерфейса пакета package и объявление тела пакета package body.

Модули проекта, в свою очередь, можно разбить на две категории: первичные и вторичные. К первичным пакетам относятся объявления пакета, объекта проекта, конфигурации. К вторичным- объявление архитектуры, тела пакета. Один или несколько модулей проекта могут быть помещены в один файл, называемый файлом проекта (design file).

Каждый проанализированный модуль проекта помещается в библиотеку проекта (design library) и становится библиотечным модулем (library unit).

Каждая библиотека проекта в языке VHDL имеет логическое имя (идентификатор).

По отношению к сеансу работы с VHDL- системой существует два класса рабочих библиотек проекта: рабочие библиотеки и библиотеки ресурсов.

Рабочая библиотека- это библиотека WORK, с которой в данном сеансе работает пользователь и в которую помещается пакет, полученный в результате анализа пакета проекта.

Библиотека ресурсов - это библиотека, содержащая библиотечные модули, ссылка на которые имеется в анализируемом модуле проекта.

В каждый конкретный момент времени пользователь работает с одной рабочей библиотекой и произвольным количеством библиотек ресурсов.

Модули, как и в обычных алгоритмических языках, - это средство выделения из ряда программ и подпрограмм общих типов данных, переменных, процедур и функций, позволяющее упростить, в частности, процесс их замены.

Так же, как в описаниях проектируемых систем разделяются описания интерфейсов и тел, в VHDL у пакета разделяются описание интерфейса и тела пакета. По умолчанию предусмотрено подключение стандартных пакетов STANDART и TEXT 10. Пакет STANDART, в частности, содержит описание булевских операций над битовыми данными и битовыми векторами. Нестандартные пакеты реализуются пользователями, желающими более точно отобразить свойства описываемых ими объектов. Например, можно в пользовательском пакете переопределить логические операции И, ИЛИ и НЕ и перейти от булевского (0, 1) к многозначному (1, 0, X? Z) алфавиту моделирования.

Проблемно- ориентированная компонента позволяет описывать цифровые системы в привычных, разработчику понятиях и терминах. Сюда можно отнести:

* понятие модельного времени now.
* данные типа time, позволяющие указывать время задержки в физических единицах
* данные вида сигнал signal, значение которых изменяется не мгновенно, как у обычных переменных, а с указанной задержкой, а также специальные операции и функции над ними
* Средства объявления объектов entity и их архитектур architecture.

Если говорить про операторную часть проблемно- ориентированной компоненты, то условно ее можно разделить на средства поведенческого описания аппаратуры (параллельные процессы и средства их взаимодействия); средства потокового описания (описание на уровне межрегистровых передач) – параллельные операторы назначения сигнала (<=) с транспортной transport или инерциальной задержкой передачи сигналов и средства структурного описания объектов (операторы конкретизации компонент с заданием карт портов port map и карт настройки generic map, объявление конфигурации и т. д.).

Параллельные операторы VHDL включают:

* оператор процесса process;
* оператор блока block;
* параллельный оператор назначения сигналу <=;
* оператор условного назначения сигналу when;
* оператор селективного назначения сигналу select;
* параллельный оператор утверждения assert;
* параллельный оператор вызова процедуры;
* оператор конкретизации компоненты port map;
* оператор генерации конкретизации generate;

Как видно из этого перечня, последовательные и параллельные операции назначения, вызова процедуры и утверждения различаются контекстно, то есть внутри процессов и процедур они последовательные, вне- параллельные.

Базовым элементом описания систем на языке VHDL является блок. Блок содержит раздел описаний данных и раздел параллельно исполняемых операторов. Частным случаем блока является описание архитектуры объекта. В рамках описания архитектуры могут использоваться внутренние, вложенные блоки. Наряду со всеми преимуществами блочной структуры программы и ее соответствия естественному иерархическому представлению структуры проекта операторы блока языка VHDL позволяют устанавливать, условия охраны (запреты) входа в блок. Только при истинности значения охранного выражения управление передается в блок и инициирует выполнение операторов его тела.

**Этапы проектных процедур с использование САПР**

**Описание проекта на языке VHDL**

**1.** Описание проекта на уровне алгоритма

Выполняется поведенческое описание, абсолютно абстрактно, не привязываясь к какому либо железу, что кстати можно делать и на C

**2.** Моделирование работы поведенческого описания устройства

Проверяем правильно ли работает придуманный нами алгоритм

**3.** Перерабатываем нашу поведенческую модель в синтезируемую и синтезируем схему

Дело в том, что не все конструкции языка можно реализовать в виде железа. Простой пример это тип Real — вещественный тип, поведенческое описания с его использованием можно выполнить и промоделировать, но для реализации в железе придется написать устройство для работы с такими числами.

\* При синтезе определенные конструкции языка реализуются в виде соответствующих им элементам цифровой электроники(мультиплексоры, логические элементы, дешифраторы и т.д.)

**4.** Проводим моделирование синтезированной модели

\* На этом этапе можно уже оценить временные задержки в разрабатываемом устройстве, так как простейшие элементы обладают определенными задержками, да и можно оценить сколько логики требуется для реализации.

**5.** Проводим размещение и трассировку на кристалле нашего проекта

\* В простейшем случае от нас потребуется связать конкретные Пины ПЛИС с портами(входами/выходами) нами разрабатываемого устройства.

\* На этом этапе синтезируемая модель виртуально размещается в кристалле на базе имеющихся в составе ПЛИС элементов цифровой электроники и происходит организация внутренних связей.

**6.** Проводится моделирования работы размещенной в ПЛИС схемы, с учетом реальных задержек в элементах ПЛИС и задержек на линиях передачи.

\* Здесь получаем достоверные временные характеристики нашего устройства проекта на ПЛИС.

\* Получаем файл для программирования ПЛИС.

**7.** Программируем ПЛИС и радуемся жизни.

Примерно так выглядит в проектирование цифровых устройств с применением ПЛИС. Как видим ничего сложного тут нет.

Теперь предлагаю попробовать написать первый проект на VHDL.

VHDL на мой взгляд язык простой, но есть свои особенности. Язык тот имеет признаки высокоурожайного, но все таки не объектноориентированный, но это компенсируется большим количеством библиотек и пакетов ( что то вроде библиотек в С).

Теперь рассмотрим основные понятия языка:

\* **Интерфейс** — описывает наше с вами устройство, как черный ящик с входами и выходами, т.е. главная его задача показать какие входы и выходы есть у нашего устройства для связи с внешним миром.( Если взять человека, то можно его как кусок мяса с портами работающими на ввод — глаза, уши и.т.д, и на вывод — гортань, руки и т.д.)

\* **Архитектура** — описывает поведение нашего устройства или раскрывает его внутреннюю структуру, т.е. в архитектуре описывается алгоритм функционирования нашего устройства. Стоит обратить внимание, что архитектура может быть описана в общем случае двумя вариантами: поведенческим стилем(описывается алгоритм работы устройства) и структурным стилем(описывается структура устройства).
Поведенческий стиль удобно использовать при описании элементов на низком уровне иерархии, а структурным на верхних уровнях иерархии, т.е. написать много маленьких устройств поведенческих стилем, а потом описать состоящие из них устройство структурным стилем, в виде связей между ними.

\* **Операторы языка** — операторы в языке бывают последовательные, а бывают параллельные. Параллельные операторы вводятся для того, чтобы отобразить параллельность протекающих в железе процессов. Но любой параллельный оператор можно заменить оператором процесса с последовательными операторами внутри его, он как раз для этого и предназначен. По-этому я бы посоветовал, тем кто переходит с контроллеров и просто программирования, использовать оператор процесса с последовательными операторами, так как это более привычно для них.

Представим. что перед нами стоит задача описать цифровое устройство заданное следующей таблицей истинности(x — входные состояния, y — выходное состояние):



Применим сначала классический метод проектирования ЦУ, построим карту Карно, получим минимизированную функцию системы и построим схему.


В результате получим два поля 1 порядка и функцию системы Y = x1\*x0 + x2'\*x0'. Апострофами обозначил отрицание.

Построим схему:



Для человека плохо разбирающегося в цифровой электронике приведенный выше процесс довольно туманный.

Теперь рассмотрим проектирование той же системы, но на языке VHDL.

1. Исходя из таблицы истинности видим, что у нас 3 входных порта и один выходной. На основе этого можем описать интерфейс нашего устройства. Описывается он при помощи ключевого слова **entity**. Синтаксис объявления интерфейса имеет следующий вид:

**entity** ID **is** [**generic** ({ID: TYPEID [:= expr];});] [**port** ({ID: **in** | **out** | **inout** TYPEID [:= expr];});] [{declaration}] [**begin** {parallel\_statement}] **end** [entity] ENTITYID; Сразу так и не врубится, но посмотрим повнимательней [] — необязательная конструкция {} — конструкция, которая может повторятся сколько угодно раз — альтернативные варианты. Пока нам требуется только знать, что все вход\выход порты описываются после ключевого слова **port**.

В нашем случае получится следующий вариант:

**entity** Sys **is port**(X: **in** bit\_vector(2 **downto** 0); y: **out** bit); **end** Sys; Что получается — у нас один порт работающий на ввод(in) и один порт работающий на вывод(out), при этом тип выходного порта bit — грубо говоря это значит, что наш порт одноразрядный, а входной порт имеет тип bit\_vector — это массив типа bit, запись вида bit\_vector(2 downto 0) обозначает 3 разрядный порт, где каждый разряд это bit. Тип bit может, является перечислимым типом и принимает два значения ('0','1').

\* На этом этапе получили черный ящик, с описанием его портов ввода/вывода.

2. На втором этапе опишем поведение нашего устройства, для этого требуется описать архитектуру. Смотрим на ее синтаксис: **architecture** ID **of** ENTITYID **is** [{declaration}] **begin** [{parallel\_statement}] **end** [architecture] ARCHID;

Как видно архитектура должна принадлежать какому-то интерфейсу и ее тело состоит из параллельных операторов. Мы используем один параллельный оператор процесса и разместим там привычные нам последовательные операторы.

Если подумать, как можно реализовать таблицу истинности, то можно прийти к выводу, что лучше всего использовать оператор который выполнял какие либо действия в зависимости от состояния переменной, в С таким оператором является **switch**(), а в VHDL **case**. Это значит, что нам потребуется рассмотреть синтаксис оператора процесса и оператора **case**.

Оператор процесса: [LABEL:] [**postponed**] **process** [( {SIGID,} )] [{declaration}] **begin** [{sequential\_statement}] **end** [**postponed**] **process** [LABEL]; Если не вдаваться в подробности, то видно что внутри процесса расположены последовательные операторы, то что нам нужно. В скобках после ключевого слова **process** указывается список чувствительности, т.е. те сигнала при изменении которых процесс запускается или срабатывает, при синтезе этот список просто игнорируется, но он необходим для правильного выполнения моделирования.

Оператор **case** [LABEL:] **case** expr **is** {**when** choice [{| choice}] **=>** {sequential\_statement}} **end case** [LABEL]; Видно, что проверяется значение сигнала или переменной — выражение *expr*, и в зависимости от его значения выполняется один из набора последовательных операторов.

Теперь попробуем собрать это все в кучу **architecture** A\_Sys **of** Sys **is begin process**(X) **begin case** X **is when** «000» => Y <= '1'; **when** «001» => Y <= '0'; **when** «010» => Y <= '1'; **when** «011» => Y <= '1'; **when** «100» => Y <= '0'; **when** «101» => Y <= '0'; **when** «110» => Y <= '0'; **when** «111» => Y <= '1'; **end case**; **end process**; **end** A\_Sys;

Вот и получили программу нашего устройства. В списке чувствительности процесса поставили X так как он должен реагировать на изменение этого порта, ну а операторе **case** просто в зависимости от комбинации на входе выдаем значение на выход, использую оператор присваивания значения сигналу **<=**.

Смотрим временные диаграммы



Структурный и поведенческий варианты описания проекта

Возможности и средства описания типовых узлов цифровой техники

Введение в язык VHDL-AMS

**Тема 2.3. Назначение и состав системных сред САПР**

[**Основные функции CAD-систем**](http://bigor.bmstu.ru/?met/?doc=190_CAD/8001.mod/?cou=140_CADedu/CAD.cou/?bck=190_CAD/8001.mod)

Функции CAD-систем в машиностроении подразделяют на функции двухмерного (2D) и трехмерного (3D) проектирования. К функциям 2D относят черчение, оформление конструкторской документации; к функциям 3D — получение трехмерных геометрических моделей, метрические расчеты, реалистичную визуализацию, взаимное преобразование 2D и 3D моделей. Трехмерные модели представляют в виде описания поверхностей, ограничивающих деталь, или указанием элементов пространства, занимаемых телом детали. Модели поверхностей сложной формы получают с помощью разновидностей *кинематического метода*, к которым относят вытягивание заданного плоского контура по нормали к его плоскости, протягивание контура вдоль произвольной пространственной кривой, вращение контура вокруг заданной оси, натягивание поверхности между несколькими заданными сечениями. В случае построения скульптурных поверхностей, проходящих через заданные точки пространства, применяют модели в форме Безье, а при требованиях высокой гладкости поверхности — модели в форме B-сплайнов. Синтез моделей сборок выполняют применением операций позиционирования и теоретико-множественных операций пересечения, объединения, вычитания к библиотечным элементам и вновь созданным моделям комплектующих деталей. В ряде систем предусмотрено также выполнение операций компоновки и размещения оборудования, проведения соединительных трасс и т.п.

К важным характеристикам CAD-систем относятся *параметризация* и *ассоциативность*. Параметризация подразумевает использование геометрических моделей в параметрической форме, т.е. при представлении части или всех параметров объекта не константами, а переменными. *Параметрическая модель*, находящаяся в базе данных, легко адаптируется к разным конкретным реализациям и потому может использоваться во многих конкретных проектах. При этом появляется возможность включения параметрической модели детали в модель сборочного узла с автоматическим определением размеров детали, диктуемых пространственными ограничениями. Эти ограничения в виде математических зависимостей между частью параметров сборки отражают ассоциативность моделей.

Параметризация и ассоциативность играют важную роль при проектировании конструкций узлов и блоков, состоящих из большого числа деталей. Действительно, изменение размеров одних деталей оказывает влияние на размеры и расположение других. Благодаря параметризации и ассоциативности изменения, сделанные конструктором в одной части сборки, автоматически переносятся в другие части, вызывая изменения соответствующих геометрических параметров в этих частях.

Корректные синтез и редактирование 3D твердотельных моделей изделий возможны с помощью нескольких методов.

Наиболее очевидный метод — задание проектировщиком изделия ограничений и условий, накладываемых на параметры модели и отражающих требования непересечения тел, соосности отверстий, компланарности, перпендикулярности и т.п.

В большинстве современных MCAD используется метод, основанный на использовании *дерева построения* модели. Деревом построения называют историю моделирования сборки, другими словами, последовательность операций создания модели, упорядоченную по времени их совершения. Согласно этому методу внесение изменений в ту или иную часть модели подразумевает переход в ту вершину дерева, которая соответствует изменяемой части, и после внесения изменений повторное выполнение всех последующих операций синтеза.

Третий способ - *синхронное моделирование*, основанное на автоматическом определении, благодаря применению экспертных систем, тех ограничений, которые в первом методе задаются пользователем. В результате упрощается работа конструктора, не требуются затраты времени на перестроение дерева модели.

**Системные среды автоматизированных систем. Подходы к интеграции ПО в САПР. Технологии интеграции ПО типа DDE и OLE. Программные средства управления проектированием в САПР.**

1 ТехнологияOLE

Технология связывания и внедрения объектов (*Object Linking and Embedding*) имеет больше функциональных возможностей, причем, если приложение поддерживаетOLE, то оно само выполняет обмен данными по этой технологии.

Операции связывания (*Linking*) и внедрения (*Embedding)*, реализованные в рамках OLE, внешне напоминают технологию DDE и обмен данными через буфер обмена. При работе по технологии OLE выполняется та же последовательность действий. Документ со встроенными OLE-объектами выглядит аналогично документу с фрагментами, вставленными через буфер обмена. Однако в этом случае при двойном щелчке мыши в поле объекта он **активизируется** и запускается приложение, в котором создавался этот объект, и в него передается объект для редактирования или выполнения других операций. После окончания работы с объектом программа-источник закрывается, а измененный объект автоматически передается обратно в документ приложения-клиента.

В рамках технологии OLE объект представляет собой сочетание данных какого-либо вида (текст, графика, видео, звук и др.) во внутреннем формате приложения-сервера, представленном в одном из стандартных форматов Windows, и информации о создавшей его программе, размере, времени создания и т.п. Таким образом, объект является законченной структурой, переносимой из одного документа в другой и сохраняющей отличительные особенности независимо от типа документа, в котором в данный момент находится.

При связывании:

      отсутствует необходимость создания второй копии объекта, что позволяет сократить требуемый объем дискового пространства;

      внесение изменений в связанный объект обеспечивает дублирование этих изменений во всех документах, с которыми объект был связан;

      запоминается путь к оригиналу, поэтому при переносе на другую машину необходимо переписать все файлы, содержащие объекты, включенные в данный документ.

При внедрении:

      изменения вставленного объекта, не отражаются в оригинале;

      вся информация хранится в одном файле и никаких проблем при переносе на другой компьютер не возникает.

В рамках OLE реализован метод **drag-and-drop** (перетащить и бросить), который обеспечивает наглядность процесса обмена данными, и его можно применять вместо операции копирования через буфер обмена даже при межоконном перемещении объектов и их частей.

OLE обеспечивает возможность местной активизации объекта – при двойном щелчке мышью объект обводится широкой штриховой рамкой, обозначающей активность, и остается на месте. Заголовок окна меняется на заголовок вызываемого приложения, а меню представляет собой комбинацию из меню приложения-источника и приложения-приемника. После выполнения операций (чаще всего, редактирования) над объектом возврат в первоначальное состояние осуществляется по щелчку мышью за пределами объекта.

Приложение-сервер и приложение-клиент обмениваются данными по наиболее новой технологии, доступной им обоим, т.е., если приложение-источник поддерживает только DDE, при работе в OLE объект будет вставлен, но возможность его активации из документа-приемника теряется.

OLE-технология, разработанная корпорацией Microsoft, обеспечивает:

    привязку – возможность вызова одной программы из другой;

    встраивание – помещение объектов, созданных в одном приложении, в документ другого.

Механизм межзадачного динамического обмена данными (Dinamical data exchange, DDE) разработан фирмой Microsoft и является составной частью операционных систем семейства Windows. Он предназначен для организации взаимодействия различных приложений (задач), одновременно выполняющихся на компьютере (существующую сетевую версию DDE здесь мы не рассматриваем). Следует отметить, что взаимодействию приложений разработчики системного программного обеспечения уделяют достаточно много внимания, более современные, чем DDE, технологии OLE и ActiveX [1] активно развиваются. Однако, несмотря на это, в прикладных программных продуктах они используются нечасто, одним из удачных примеров таких программ можно назвать “Эффект Офис” фирмы Гарант-Практик, где реализован просмотр и редактирование документов с помощью “родных” для них приложений, вызываемых через OLE, а с помощью DDE происходит передача команд в “Эффект Офис” из Microsoft Word и Microsoft Excel.

При этом, сами по себе достаточно многие приложения поддерживают указанные механизмы, однако для многих разработчиков проще самим реализовать их возможности, чем воспользоваться ими. По-видимому, главным препятствием здесь служит недостаток или полное отсутствие документации с описанием возможностей и методов реализации механизмов обмена конкретных программных продуктов. Что же касается популярных книг, посвященных общим вопросам философии и, изредка, реализации обмена информацией между задачами, ситуация несколько отличается: довольно много книг про OLE [1-3] и совсем нет про DDE (так, в [3,4] есть лишь упоминания о его существовании). В результате, автору при разработке системы Евфрат пришлось изучить OLE по указанным книгам и, используя примеры из них, сделать простейшие программы, а затем чисто экспериментально изучать возможности работы с Бэйсиком MSWord через OLE. Точно также, руководствуясь описаниями и примерами из электронной системы помощи Word, Excel и Microsoft Developer Studio (MSDEV) и, частично, [5], удалось разобраться в организации DDE.

Сравнительный анализ DDE и OLE не входит в намерения автора, поэтому ограничимся лишь несколькими замечаниями. Представляется, что использование DDE предпочтительнее OLE (там, где это возможно, естественно). Это обусловлено тем, что сам механизм DDE проще и прозрачнее организован, для его реализации требутся гораздо меньше исходной информации о программе, с которой организуется взаимодействие. Далее, DDE организован на сообщениях Windows, которые могут быть обработаны самим разработчиком и отслежены средствами операционной системы, что существенно облегчает отладку программ, в то время как OLE такой возможности не дает. Кроме того, механизм OLE представляется менее надежным, т.к. в распоряжении автора имеются конкретные примеры некорректной работы с OLE одной из наиболее популярных программ.

Механизм DDE был реализован автором при разработке системы Евфрат для взаимодействия с MSWord в части выполнения Евфратом некоторых действий (открытие документа, сохранение, поиск, показ свойств документа) по командам пользователя, работающего в среде MSWord. При этом использовалась библиотека Microsoft Foundation Classes (MFC), предоставляющая разработчику широкий выбор функций и классов C++, существенно облегчая тем самым разработку приложений Windows. В частности, здесь нас будут интересовать оконные классы, иерархия которых охватывает значительную часть разумных потребностей разработчика. Базовым для них является класс CWnd, описывающий простейшее окно Windows, в котором реализованы основные его свойства, из которых для нас существенна способность отправлять, принимать и обрабатывать сообщения Windows.

 2. Краткое описание протокола DDE.

 Реализация механизма динамического обмена данными состоит в программировании определенной последовательности действий, описываемой соответствующим протоколом обмена, при описании его будем следовать электронной системе помощи MSDEV (Help MSDEV).

О двух приложениях, ведущих обмен данными, говорят как о приложениях, участвующих в диалоге DDE, причем то из них, которое начало его, называется клиентом, а откликнувшееся на запрос клиента - сервером. Естественно, одно приложение может участвовать в нескольких таких диалогах, выступая в одних сервером, а в других клиентом. Диалог DDE происходит между двумя окнами, одно на каждое приложение, причем статус этого окна не важен: îíî ìîæåò áûòü ãëàâíûì îêíîм, окном, связанным с открытым документом, или просто отдельным невидимым окном, единственной задачей которого и является обслуживание диалога. Протокол DDE требует, чтобы каждое окно участвовало только в одном диалоге, если же требуется, чтобы приложение участвовало в нескольких, оно должно создать соответствующее число дополнительных окон.

DDE идентифицирует передаваемые данные по именам приложения, темы и пункта (application, topic, item), причем приложение и тема однозначно определяют диалог, а пункт характеризует передаваемые данные. Имя приложения - это обычно имя главного окна или его exe-файла, тема - обобщенное имя большого набора данных, состоящего из отдельных пунктов, во многих случаях это имя текстового файла, загруженного в окно, пункт - название порции данных для обмена, клиент может получать эти данные у сервера, а также изменять их, передавая через DDE. Данные могут передаваться в любом формате, допускаемом для передачи через буфер обмена (clipboard), стандартном или зарегистрированном пользователем. Считается, что каждое приложение должно иметь одну специально выделенную тему (System), где содержится справочная информация о нем, доступная для других задач. Здесь могут содержаться в текстовом формате список форматов обмена, краткая информция о командах, выполняемых сервером, список пунктов темы System, список тем, доступных в данный момент для диалога, а также другая полезная информация. Данные при обмене передаются через специально аллоцированные области памяти, причем детально оговаривается, какое из приложений и в какой момент освобождает эту память.

При обмене DDE используются сообщения:

|  |  |
| --- | --- |
| WM\_DDE\_ACK | Подтверждение получения сообщения |
| WM\_DDE\_DATA | Посылка данного приложению-клиенту по его запросу. |
| WM\_DDE\_EXECUTE | Посылка серверному приложению текстовой строки, которая рассматривается как последовательность команд для выполнения |
| WM\_DDE\_INITIATE | Инициализация диалога DDE |
| WM\_DDE\_POKE | Посылка данного приложению-серверу с изменением содержимого указанного пункта |
| WM\_DDE\_REQUEST | Запрос приложению-серверу на получение данного из указанного пункта. |
| WM\_DDE\_TERMINATE | Завершение диалога |
| WM\_DDE\_ADVISE, WM\_DDE\_UNADVISE | Включение и выключение режима, в котором серверное приложение извещает клиентское об изменении данных из указанного пункта. |

Все эти сообщения должны посылаться функцией **PostMessage,** кроме WM\_DDE\_INITIATE и WM\_DDE\_ACK в ответ на него, которые посылаются функцией **SendMessage,** во всех сообщениях первым параметром wParam передается дискриптор окна, пославшего его.

 **Инициализация диалога** состоит в обмене сообщениями, во время которого клиент и сервер запоминают дискрипторы окон, участвующих в диалоге и, возможно, выполняют какие - либо еще действия (например, инициализацию переменных). Она делается с помощью сообщения WM\_DDE\_INITIATE, второй параметр сообщения

 lParam = MAKELPARAM(atomApp, atomTopic),

 где atomApp и atomTopic - “атомы”, содержащие имена приложения и темы сервера, с которам клиент намерен вступить в диалог или нули, если они заранее неизвестны. “Атомы” - это специфические объекты Windows, которые создаются функцией **GlobalAddAtom**, уничтожаются функцией **GlobalDeleteAtom и** представляют собой аллоцированную область памяти, содержащую текст. Это сообщение может быть послано определенному окну, если оно известно, либо всем главным окнам приложений, для чего при посылке используется дискриптор равный -1. Если окно, получившее сообщение, готово включиться в диалог с пославшим его, оно отвечает посылкой сообщения WM\_DDE\_ACK, во втором параметре которого указываются имена приложения и темы сервера, как показано выше. Для случая, когда сервер и (или) тема явно не указаны, эта информация совершенно необходима, иначе она дает дополнительные возвожности для контроля.

**Передача данных указанного пункта.** После того, как диалог DDE установлен, клиентское приложение может либо получить содержимое пункта, используя сообщение WM\_DDE\_REQUEST, либо изменить лежащие там данные, заслав их туда сообщением WM\_DDE\_POKE. Для получения необходимо указать название пункта и формат передачи данных, передав их во втором параметре сообщения lParam. Указанная информация кодируется функцией PackDDElParam, работа которой зависит от сообщения, для которого готовится параметр:

lParam = PackDDElParam(WM\_DDE\_REQUEST, CF\_TEXT, atomItem).

В этом примере требуется передать в текстовом формате содержимое пункта, определяемого “атомом”, содержащем его имя. Если сервер может передать требуемые данные в указанном виде, он аллоцирует память с флагом GMEM\_MOVEABLE | GMEM\_DDESHARE для структуры DDEDATA, копирует туда данные и посылает клиенту сообщение WM\_DDE\_DATA, lParam которого формируется как показано в примере:

lParam = PackDDElParam(WM\_DDE\_DATA, (UINT) hData, atomItem),

где hData - дискриптор глобальной памяти для структуры DDEDATA. Если же требуемые данные не могут быть переданы, сервер посылает в ответ отрицательное сообщение WM\_DDE\_ACK с

lParam = PackDDElParam(WM\_DDE\_ACK, 0, atomItem).

В случае успешной отправки информации сервер может запросить у клиента (установив соответствующий флажок в структуре DDEDATA) ответ о получении и обработке переданных им данных. Это делается посылкой отрицательного сообщения WM\_DDE\_ACK либо положительного с

lParam = PackDDElParam(WM\_DDE\_ACK, 0x8000, atomItem).

 Засылка данных на сервер аналогична передаче их сервером по запросу клиента. Разница заключается в том, что в этом случае используется сообщение

WM\_DDE\_POKE и структура DDEPOKE, а ответ сообщением WM\_DDE\_ACK (положительным или отрицательным в случае невозможности обработать переданные данные) обязателен.

 **Передача команд серверу**, которые могут быть им выполнены, делается сообщением WM\_DDE\_EXECUTE с

lParam = PackDDElParam(WM\_DDE\_EXECUTE, 0, (UINT) hCommand),

где hCommand - дискриптор глобальной памяти, аллоцированной с флагом GMEM\_MOVEABLE | GMEM\_DDESHARE, куда занесена текстовая строка с командой. В ответ сервер посылает клиенту положительное или отрицательное сообщение WM\_DDE\_ACK, причем в lParam снова передается hCommand, что может быть использовано для контроля за прохождением команд. При этом отрицательный ответ дается сразу после выяснения невозможности выполнить команду, а положительный - только после ее выполнения. Единственным исключением из этого правила является случай, когда выполнение команды приводит к закрытию серверного приложения. В такой ситуации следует сразу ответить положительным WM\_DDE\_ACK, послать WM\_DDE\_TERMINATE для завершения диалога, а потом уже закрыть приложение.

 **Завершение диалога DDE** делается путем обмена сообщениями WM\_DDE\_TERMINATE, где lParam=0, а wParam, как обычно, содержит дискриптор окна, пославшего сообщения. О прекращении диалога может объявить как серверное, так и клиентское приложение, так что оба они должны быть готовы принять и обработать это сообщение, приложение всегда должно прекратить все диалоги перед закрытием. Получив сообщение WM\_DDE\_TERMINATE , приложение должно тут же ответить также сообщением WM\_DDE\_TERMINATE , а не WM\_DDE\_ACK, как обычно. Посылка такого сообщения окном означает, что в дальнейшем оно не будет посылать сообщения и получатель может закрыть свое окно, участвующее в диалоге. В тоже время в случае получения сообщения (видимо, ошибочного) от окна, прекратившего диалог, не следует отвечать на него, т.к. оно может быть уже уничтожено.

 2. Особенности реализации протокола DDE в системе Евфрат.

В процессе работы по реализации протокола было выявлено несколько ошибок в Help MSDEV. Самой неприятной особенностью этих ошибок оказалось то, что в среде Windows-95 программа работала абсолютно нормально, тогда как под Windows NT работать отказывалась.

 Эти ошибки содержатся в примерах посылки сообщений DDE:

 1. При описании сообщения WM\_DDE\_TERMINATE приведен пример посылки этого сообщения в виде

PostMessage(hwndServerDDE, WM\_DDE\_TERMINATE,

 PackDDElParam(WM\_DDE\_TERMINATE, (UINT) hwndClientDDE, 0), 0);

Это означает, что первый параметр, wParam получается запаковкой функцией PackDDElParam дискриптора hwndClientDDE окна, пославшего сообщение, в то время как он должен передаваться через wParam непосредственно. По-видимому, в Windows-95 эти действия дают один результат, а в Windows NT разные.

 2. При описании посылки данных с помощью сообщения WM\_DDE\_DATA рекомендуется параметр lParam (см. выше) получать следующим образом:

lParam = PackDDElParam(WM\_DDE\_ACK, (UINT) hData, atomItem),

т.е. неверно указано сообщение, для которого готовится lParam. Видимо, как и в предыдущем случае, функция PackDDElParam по-разному работает в разных операционных системах.

 К сожалению, нельзя гарантировать, что приведенными примерами список ошибок исчерпывается.

 Для реализации функций DDE написан оконный класс (в смысле языка C++) TDdeWindow, производный от базового класса MFC CWnd (см. выше). Предусматривается, что окна, описываемые объектами этого класса, будут невидимыми, а предками их (в смысле Windows) будут либо главное окно приложения (в случае, когда пишется клиентское приложение или тема System для серверного), либо окна для просмотра файлов (для получения данных из них при работе в качестве сервера). В соответствии с идеологией MFC, к стандартному списку обрабатываемых сообщений базового класса добавлены функции обработки описанных выше сообщений DDE, кроме того в этот класс включены функции, реализующие клиентские возможности

созданного окна, так что один и тот же класс используется для создания окон как клиентских, так и серверных приложений.

 Рассмотрим более подробно работу клиентской части. Она начинается с создания невидимого окна и одновременно объекта класса TDdeWindow, а завершается рассылкой всех необходимых сообщений при уничтожении его, т.е. при обработке соответствующего сообщения Windows. Это происходит автоматически при уничтожении окна-предка, так что не требуется вызов специальных функций для закрытия диалога. В случае, если диалог разрывается по требованию партнера, дискриптор его окна просто убирается из списка окон, участвующих в диалоге.

 Для начала работы в качестве клиента с прилжением-сервером служит функция

int DdeInit(const char \*cApp, const char \*cTopic),

параметрами которой являются имена приложения и темы, а возвращаемое целое число определяет “канал” (обозначается iChannel), используемый при дальнейших вызовах функций и идентифицирующий окно сервера, участвующее в диалоге. Для содержательной работы с сервером служат функции

 int DdeExec(int iChannel, const char \*cCommand);

 int GetDdeRequest(int iChannel, const char \*cItem, CString &sRet);

 int DdePoke(int iChannel, const char \*cItem, const char \*pSent);

Первая из них позволяет передать командную строку для выполнения, результат этого действия (нормальное выполнение команды или отказ ее выполнить) определяется по величине возвращаемого значения, как, впрочем, и для других функций. Вторая и третья функции служат для получения и отправки данных серверу в текстовом виде, параметр cItem определяет пункт, sRet - строка для хранения полученного данного (Cstring - класс MFC, описывающий строковые переменные), pSent - указатель на отправляемый текст. И, наконец, для разрыва диалога, если это необходимо, есть функция

 int DdeTerm(int iChannel);

 Работа этих функций состоит в подготовке данных, заполнении соответствующих структур и посылке сообщений. Однако, для выяснения результата требуется обработка ответного WM\_DDE\_ACK, а для получения переданного сервером данного - WM\_DDE\_DATA, что внутри этих функций невозможно, т.к. сообщения Windows обрабатываются функциями, вызываемыми после их прихода. Поэтому в функциях клиентской части после посылки сообщений запускается вторичный цикл обработки сообщений, выход из которого происходит либо по истечении указанного времени, либо после получения ответа, что определяется по установке или сбросу соответствующего флажка. Таким образом, возвращаемое значение определяется по результатам обработки ответных сообщений, а если они не пришли за отведенное время, выдается сообщение об ошибке.

 Работа приложения в качестве сервера начинается, как и в клиентском случае, созданием невидимого окна и объекта класса TDdeWindow. Основным отличием его от клиентского является, естественно, способность начинать диалог по запросу, а также выполнять передаваемые ему команды, принимать данные и отправлять их по запросу в нужном виде. Для реализации первого свойства организована специфическая иерархия серверных окон. Как уже говорилось, окно DDE всегда является потомком какого-либо другого, поэтому оно само не может получить сообщение о начале диалога, рассылаемое главным окнам приложений. В связи с этим обработчик WM\_DDE\_INITIATE включается в класс главного окна и создается потомок главного окна приложения, главное окно DDE, которое хранит информацию о всех создаваемых позже серверых окнах. Далее главное окно приложения получает сообщение о приглашении и просто посылает его главнму окну DDE, которое, в свою очередь, рассылает его всем остальным. Если какое-то из окон обнаружит, что имена приложения и темы в сообщении-приглашении соответствуют его параметрам, оно откликается сообщением WM\_DDE\_ACK.

Для создания главного окна DDE вызывается функция

 BOOL CreateDdeServer(DdeFunc pFunc, const char \*AppName=NULL),

первым параметром которой является указатель на написанную разработчиком функцию для выполнения остальных серверных действий, которая определяется как

typedef BOOL (\*DdeFunc)(UINT iMessage, const char \*String,

 TDdeWindow \*pThisWnd);

Ее параметры - идентификатор сообщения WM\_DDE\_EXECUTE и т.д., указатель на текстовую строчку, которая содержит либо команду, либо имя пункта, а также указатель на окно, которое вызывает функцию (для вызова его собственных функций, если потребуется). Функция CreateDdeServer создает окно с темой System, имя приложения можно задать явно (второй параметр) либо использовать умолчательное.

 Следующие серверные окна создаются функцией

 BOOL AddDdeServer(DdeFunc pFunc, const char \*TopicName, CWnd \*pParent);

в которую опять передается указатель на DdeFunc, имя темы (имя приложения уже задано), указатель на окно-предка, имена пунктов задаются только внутри функций DdeFunc.

 В некоторых случаях при выполнении команды в качестве сервера может появиться потребность послать ответное сообщение клиенту до фактического завершения ее выполнения. Это связано с тем, что клиент может выдать сообщение об ошибке, не получив ответа за определенный срок (примерно минута для MS Word), а выполнение команды на сервере может быть длительным (например, для Евфрата это индексация документа или работа в открытом по команде из MS Word диалоге открытия или поиска). Для посылки такого сообщения в класс TDdeWindow включена функция

 void FreeClientInExec(BOOL bRes);

посылающая положительное или отрицательное (определяется bRes) WM\_DDE\_ACK клиентскому окну. Естественно, пользоваться этой возможностью надо очень осторожно, принимая меры к тому, чтобы следующая команда от клиента не поступила до фактического выполнения сервером предыдущей (возможно также организовать очередь команд на сервере). Видимо, реально это достижимо только при совместной разработке клиентской и серверной части.

**Примеры подсистем управления данными и проектированием.**

[**Автоматизированное управление технологическими процессами**](http://bigor.bmstu.ru/?met/?doc=020_ERP/erp201.mod/?cou=140_CADedu/CAD.cou/?bck=020_ERP/erp201.mod)

В *автоматизированных системах управления технологическими процессами*, часто называемых системами промышленной автоматизации, можно выделить свои иерархические уровни.

На верхнем (диспетчерском) уровне АСУТП осуществляются сбор и обработка данных о состоянии оборудования и протекании производственных процессов для принятия решений по загрузке станков и выполнению технологических маршрутов. Эти функции возложены на систему диспетчерского управления и сбора данных, называемую SCADA (Supervisory Control and Data Acquisition). Кроме диспетчерских функций, система SCADA выполняет роль инструментальной системы разработки программного обеспечения (ПО) для промышленных систем компьютерной автоматизации.

На уровне управления технологическим оборудованием (на уровне контроллеров) в АСУТП выполняются запуск, тестирование, выключение станков, сигнализация о неисправностях, выработка управляющих воздействий для рабочих органов программно управляемого оборудования. Для этого в составе технологического оборудования используются системы управления на базе программируемых контроллеров — компьютеров, встроенных в технологическое оборудование. Поэтому системы промышленной автоматизации часто называют встроенными системами (*Embedded Computing System*), а используемые при этом компьютеры - *промышленными компьютерами*.

Техническое обеспечение АСУТП представлено персональными ЭВМ и микрокомпьютерами (программируемыми контроллерами), распределенными по контролируемым участкам производства и связанными друг с другом с помощью шин.

На верхнем уровне иерархии шин осуществляется связь компьютеров системы SCADA и серверов баз данных, здесь используются технологии локальных вычислительных сетей, как правило, сети Ethernet (рис. 5) Для связи компьютеров с высокоскоростными периферийными устройствами служат шины Infiniband, Fiber Channel, USB, FireWire 1394, с низкоскоростными устройствами связь осуществляют через интерфейсы RS-232, RS-422, RS-485.

|  |
| --- |
| F:\SAPO\230113 Komputernye Systemy\Sapr TEORIA\UCHEBNIK SAPR\6 Автоматизированные системы в промышленности\Автоматизированное управление технологическими процессами_files\saved_resource(10) |

Рисунок 5.  Архитектура АСУТП

На среднем уровне (уровне контроллеров) АСУТП для связи компьютеров с системами ЧПУ обычно применяют сети Fieldbus — так называемые полевые шины. Особенностями Fieldbus являются режим реального времени, детерминированность поведения, повышенная надежность при работе в промышленной среде. В полевых шинах имеют место протоколы трех уровней: физического уровня, канального уровня, прикладного уровня. Примерами полевых шин являются последовательные шины Profibus, Interbus/S, CANbus. Всего на рынке имеется несколько десятков типов таких шин.

На нижнем уровне АСУТП соединение модулей контроллеров, датчиков, измерительного и другого оборудования в пределах одного функционального узла (например, соединение слотов в крейте) выполняется посредством магистрально-модульных параллельных шин, таких как VMEbus, CompactPCI, или последовательных шин типа Infiniband или CompactPCI Express.

Программное обеспечение АСУТП представлено операционными системами, программами SCADA, драйверами и прикладными программами контроллеров.

**Тема 2.4. Проектирование радиоэлектронных средств САПР в P-CAD**

[**Схемотехническое проектирование**](http://bigor.bmstu.ru/?met/?doc=010_EDA/eda100.mod/?cou=140_CADedu/CAD.cou/?bck=010_EDA/eda100.mod)

Назначением *схемотехнического проектирования* является разработка принципиальных электрических схем СБИС и РЭА. Задачи схемотехнического проектирования разделяются на задачи анализа и синтеза проектных решений.

Процедуры схемотехнического проектирования обычно непосредственно не входят в маршрут проектирования СБИС. При проектировании интегральных схем их применяют, в основном, при отработке библиотек функциональных компонентов СБИС. Но при разработке принципиальных электрических схем радиоэлектронных устройств в различных приложениях они могут стать основными проектными процедурами (наряду с конструкторским проектированием печатных плат).

Схемотехнический анализ может потребоваться также и внутри маршрута проектирования СБИС. Так, в случае МОП-схем появляется промежуточный уровень абстракции (switch level) между схемотехническим и вентильным уровнями, на котором элементами моделей являются не вентили, а МОП-транзисторы. Благодаря представлению последних в виде переключателей удается отобразить процессы в схеме более детально, чем с помощью программ логического моделирования

В области автоматизации схемотехнического проектирования наибольшее распространение получили варианты программы Spice, созданные в нескольких фирмах.

В программе *PSpice* для персональных компьютеров предусмотрены статический, динамический и частотный виды анализа, смешанное логико-аналоговое моделирование (mixed-signal simulation), температурный (с индивидуальными значениями температуры по приборам) и шумовой анализы, расчеты в наихудшем случае и статистический по методу Монте-Карло, спектральный анализ. В логической части реализовано событийное моделирование, выявляются риски сбоя, рассчитываются зависимые от нагрузки задержки. Для оптимизации параметров, выполняемой градиентными методами, можно использовать программу PSpice Optimizer.

В библиотеках программы PSpice имеется несколько тысяч математических моделей элементов (диодов, биполярных и полевых транзисторов, операционных усилителей, стабилизаторов, тиристоров, компараторов, магнитных устройств с учетом насыщения и гистерезиса, оптронов, кварцевых резонаторов, длинных линий с учетом задержек, отражений, потерь и перекрестных помех, и др.) Библиотека открыта для включения моделей пользователя, имеются соответствующие инструментальные средства пополнения библиотеки. Предусмотрено взаимодействие аналоговой и цифровой частей схемы.

Известен также ряд других программ аналогового и смешанного моделирования. В их число входят программы ICAP/4Window (Intusoft), Saber Mixed-technology Simulator (фирмы Analogy), Viewanalog (Viewlogic Systems), Continuum (Mentor Graphics) и др.

К числу отечественных программ схемотехнического анализа относятся программа AVOCAD компапнии Юник Айсиз и программы серии ПА - это ПА7, в которой наряду с видами анализа, обычными для программ анализа электронных схем, реализовано моделирование механических, гидравлических, тепловых процессов, и ПА9, написанная на языке Java.

Схемотехническое проектирование радиотехнических схем (RF-схем) отличается рядом особенностей математических моделей и используемых методов, особенно в области СВЧ-диапазона. Для анализа линейных схем обычно применяют методы расчета полюсов и нулей передаточных характеристик. моделирование стационарных режимов нелинейных схем чаще всего выполняют с помощью методов пристрелки или гармонического баланса. *Гармонический баланс* основан на разложении неизвестного решения в ряд Фурье, его подстановки в систему дифференциальных уравнений с группированием членов с одинаковыми частотами тригонометрических функций, в результате получают системы нелинейных алгебраических уравнений, подлежащие решению. Сокращение времени в случае слабо нелинейных схем достигается при моделировании СВЧ-устройств с помощью рядов Вольтерра. Анализ во временной области для ряда типов схем выполняют с помощью программ типа Spice путем интегрирования СОДУ.

Имеются специальные программы для анализа электромагнитной совместимости компонентов в конструктивах РЭА. К ним, например, относятся программы семейства OmegaPLUS, с помощью которых определяется форма сигналов в конструкциях с печатными платами, кабельными соединениями, микрополосковыми линиями; анализируются статические электрические и магнитные поля в геометрических плоских и объемных конструкциях; выполняется расчет полосковых и микрополосковых устройств, взаимных индуктивностей и емкостей многопроводных линий передачи; моделируются электромагнитные излучения в печатных платах; рассчитываются задержки с учетом паразитных емкостей и индуктивностей При моделировании компоненты схемы представляются в виде линейных эквивалентных схем входных и выходных цепей, проводится частотный анализ, фиксируются максимальные амплитуды напряженностей электрического и магнитного полей, электрических токов и напряжений, результаты используются для принятия необходимых конструктивных решений

# Автоматизация проектирования электроники

Автоматизация проектирования электроники (англ. *Electronic Design Automation, EDA*) — категория программных инструментов для проектирования и производства электронных систем — от печатных плат до интегральных микросхем, — являющаяся разновидностью автоматизированного проектирования ([CAD](http://plmpedia.ru/wiki/CAD)). Термин EDA зачастую используется также для обозначения систем автоматизированного конструирования ([CAE](http://plmpedia.ru/wiki/CAE)) и производства ([CAM](http://plmpedia.ru/wiki/CAM)) в области электроники — наряду с термином ECAD (англ. *Electronic Computer-Aided Design*), проектирование электроники с помощью компьютера.

На рынке оперируют более 30 вендоров. Годовой доход шести крупнейших из них в 2011:

$4.0 млрд. Synopsys $2.94 млрд. Cadence $1.42 млрд. Mentor Graphics $489 млн. Magma Design Automation $8.29 млн. SpringSoft. В сумме составляет более $10 млрд.

**Этапы проектирования. Назначение САПР P-CAD**

[**Проектирование печатных плат**](http://bigor.bmstu.ru/?met/?doc=010_EDA/eda041.mod/?cou=140_CADedu/CAD.cou/?bck=010_EDA/eda041.mod)

Среди программного обеспечения *проектирования печатных плат* для платформы Wintel хорошо известны система OrCAD, программы SPECCTRA и PCB Design Studio (Cadence Design Systems), системы P-CAD и Protel 99SE (Protel International), а также некоторые другие.

После объединения в 1999 г. компаний OrCAD и Cadence Design Systems система моделирования и сквозного проектирования аналого-цифровых электронных устройств OrCAD продолжает развиваться и в настоящее время представлена своей версией OrCAD 9.2.

С помощью ряда редакторов, имеющихся в OrCAD, выполняется интерактивное проектирование печатных плат. Имеются программы размещения компонентов, автотрассировки проводников и создания управляющих файлов для фотоплоттеров. В состав системы входят также средства для анализа и оптимизации электронных схем и проектирования устройств на ПЛИС. Поэтому система OrCAD признана, как система сквозного проектирования РЭА.

Программа SPECCTRA компании Cadence — одна из наиболее мощных программ проектирования печатных плат — может выполнять размещение и трассировку как в интерактивном, так и в автоматическом режиме. Размещение происходит за несколько проходов, во время которых выявляются и устраняются конфликты типа пересечений проводников в одном слое или нарушения проектных норм.

Широко известна система проектирования печатных плат P-CAD. Версия этой системы, именовавшаяся Accel EDA 15.0, после слияния компаний Accel Technologie и Protel International получила название P-CAD. С ее помощью выполняют полный цикл проектирования печатных плат, включая интерактивное размещение компонентов, трассировку проводников и выпуск документации. Автоматическое размещение компонентов на плате и автоматическая трассировка проводников осуществляются с помощью отдельно поставляемого пакета SPECCTRA.

Проектирование печатной платы в системе P-CAD 2004 в рекомендуемом типовом маршруте проектирования, как правило, начинается с настройки графического редактора Schematic и ввода с его помощью принципиальной электрической схемы. Ввод начинается с размещения на рабочем поле символов компонентов. Выводы компонентов соединяются проводниками. Отредактированная схема проверяется на наличие ошибок и создается список компонентов и соединений для передачи в редактор печатных плат.

Список соединений, созданных в схемном редакторе загружается в графический редактор РСВ, на рабочем поле появляются группы компонентов с индикацией электрических связей между ними.

Далее следует ручное размещение компонентов на печатной плате. Раскладка проводников и металлизированных областей может проводиться в ручном, интерактивном или автоматическом режимах. После трассировки проект проверяется на наличие ошибок и нарушений технологических норм и при необходимости редактируется.

На заключительном этапе готовятся файлы для изготовления шаблонов и сверления, и проект передается в производство.

В системе P-CAD имеются библиотека корпусов микросхем, библиотека современных импортных электрорадиоэлементов, которую можно пополнить библиотеками отечественной элементной базы, препроцессоры подготовки информации для изготовления фотошаблонов в форматах ряда известных фотоплоттеров. Поддержка текстовых форматов DXF и PDIF позволяет обмениваться информацией с такими распространенными пакетами, как AutoCAD, OrCAD, Viewlogic и др.

Компания Protel International (новое название Altium) предлагает наряду с P-CAD 2000 систему сквозного проектирования РЭА Protel 99SE собственной разработки, имеющую возможности, аналогичные возможностям системы P-CAD 2000.

Переход от системы CAD к системе CAM выражается в преобразовании результатов конструкторского проектирования в управляющую информацию для генераторов изображений. В случае печатных плат для такого перехода можно использовать программы семейства LAVENIR фирмы Lavenir Technology, с их помощью создаются и дорабатываются управляющие файлы для фотоплоттеров и станков с ЧПУ, контролируется соблюдение проектных норм.

Технологическое проектирование печатных плат заключается в преобразовании результатов конструкторского проектирования в файлы управляющей информации для фотоплоттеров и сверлильных станков с ЧПУ.

Рассмотрим маршрут проектирования печатных плат, применяемый в системе P-CAD и который можно считать типовым.

Проектирование начинается с ввода принципиальной электрической схемы (размещение на рабочем поле условных графических обозначений компонентов и линий связи компонентов) в графическом схемном редакторе. Проектирование собственно печатной платы производится конструктором в графическом редакторе РСВ, к которому предварительно подключаются нужные библиотеки. Производится упаковка схемы на печатную плату. Загружается список соединений. В ручном режиме размещаютя компоненты на поверхности печатной платы. Раскладка проводников и металлизированных областей может выполняться также в интерактивном или автоматическом режимах. Выполняются поиск и при необходимости устранение ошибок и нарушений технологических норм. Далее готовятся файлы с данными для производства печатной платы.

Другой пример маршрута проектирования печатных плат - маршрут PADS, представляемый компанией MENTOR GRAPHICS.

**Установка метрических параметров проекта.**

**Индивидуально для каждой программы в СВОЙСТВАХ ПРОЕКТА.**

***Практические работы:***

Создание символьного изображения РЭК.

Создание конструкторско-технологического образа РЭК.

Формирование библиотеки радиоэлектронного компонента.

Формирование принципиальной электрической схемы.

Создание конструктива ПП, упаковка данных и размещение РЭК на ПП.

Трассировка соединений.

***На практических занятиях используются САПР AutoCad, PCad, Altium, ТехноПро.***