Оглавление

[Ответы на вопросы 2](#_Toc282748770)

[1. Краткая характеристика процесса проектирования 3](#_Toc282748771)

[2. Определение САПР 3](#_Toc282748772)

[3. Виды обеспечения САПР 3](#_Toc282748773)

[4. Принципы системного подхода к процессу проектирования 4](#_Toc282748774)

[5. Общий подход к делению проектирования 4](#_Toc282748775)

[6. Деления процесса проектирования по временному признаку 5](#_Toc282748776)

[7. Деление процесса проектирования по характеру выполняемых работ 5](#_Toc282748777)

[8. Деление процесса проектирования по блочно-иерархическому подходу 6](#_Toc282748778)

[9. Блочно-иерархический подход к процессу проектирования. 6](#_Toc282748779)

[10. Типовые проектные процедуры. 7](#_Toc282748780)

[11. Задача синтеза в процессе проектирования. 7](#_Toc282748781)

[12. Задача анализа в процессе проектирования. 8](#_Toc282748782)

[13. Задача оптимизации в процессе проектирования. 8](#_Toc282748783)

[14. Математическая постановка задачи оптимизации. 9](#_Toc282748784)

[15. Общая характеристика критериев оптимизации. 10](#_Toc282748785)

[16. Аддитивный критерий оптимизации. = 15 11](#_Toc282748786)

[17. Мультипликативный критерий оптимизации. 11](#_Toc282748787)

[18. Минимаксные критерии оптимизации. 11](#_Toc282748788)

[19. Частные критерии оптимизации. 11](#_Toc282748789)

[20. Методы задания предпочтений на множестве частных критериев в задаче оптимизации. 12](#_Toc282748790)

[21. Общая характеристика системного уровня проектирования. 12](#_Toc282748791)

[22. Задачи синтеза и анализа на системном уровне проектирования.(мб это не то) 13](#_Toc282748792)

[23. Общая характеристика СМО. 14](#_Toc282748793)

[24. Аналитическая модель СМО. 15](#_Toc282748794)

[25. Имитационная модель СМО. 16](#_Toc282748795)

[26. Динамическая структура моделирования СМО. 17](#_Toc282748796)

[27. Событийный метод моделирования СМО. 17](#_Toc282748797)

[28. Языки моделирования, ориентированные на описание событий. 19](#_Toc282748798)

[29. Языки моделирования, ориентированные на описание процессов. 19](#_Toc282748799)

[30. Краткое описание ASPOL. 20](#_Toc282748800)

[31. Краткое описание GPRS (GPSS). 21](#_Toc282748801)

[32. Пример аналитической модели СМО. 21](#_Toc282748802)

[33. Пример имитационной модели СМО. 22](#_Toc282748803)

[34. Общая характеристика уровня функционально-логического проектирования. 23](#_Toc282748804)

[35. Задача синтеза на уровне функционально-логического проектирования. 24](#_Toc282748805)

[36. Задача анализа на этапе функционально-логического проектирования. 24](#_Toc282748806)

[37. Модели элементов в системе логического моделирования. 25](#_Toc282748807)

[38. Модели сигналов в системе логического моделирования. 25](#_Toc282748808)

[39. Модели схем в системе логического моделирования. 26](#_Toc282748809)

[40. Риски сбоя в схемах и способы их обнаружения 26](#_Toc282748810)

[41. Общая характеристика алгоритмов моделирования. 26](#_Toc282748811)

[42. Алгоритм простой итерации (2-ая модель сигналов). 27](#_Toc282748812)

[43. Алгоритм ускоренной итерации (2-ая модель сигналов). 29](#_Toc282748813)

[44. Алгоритм простой итерации (3-ая модель сигналов). 31](#_Toc282748814)

[45. Алгоритм ускоренной итерации (3-ая модель сигналов). 31](#_Toc282748815)

[46. Алгоритмы ранжирования. 31](#_Toc282748816)

[47. Событийный алгоритм моделирования (статическая модель элементов). 33](#_Toc282748817)

[48. Событийный алгоритм моделирования (ЛД – модель элемента). 35](#_Toc282748818)

[49. Параллельная реализация итерационных алгоритмов. 35](#_Toc282748819)

[50. Последовательностная реализация событийных алгоритмов. 35](#_Toc282748820)

[51. Понятия «контроль» и «диагностика» в теории диагностирования СВТ. Методы контроля и диагностики. 35](#_Toc282748821)

[52. Избыточность и трудоемкость в процедурах контроля и диагностирования СВТ. 36](#_Toc282748822)

[53. Общая характеристика функционального контроля СВТ. 36](#_Toc282748823)

[54. Общая характеристика параметрического контроля СВТ. 38](#_Toc282748824)

[55. Общая характеристика тестового контроля СВТ. 39](#_Toc282748825)

[56. Контроль передачи информации. 40](#_Toc282748826)

[57. Метод таблиц истинности синтеза тестов. 41](#_Toc282748827)

[58. Метод активизации путей синтеза тестов. 41](#_Toc282748828)

[59. Д-алгоритм синтеза тестов. 42](#_Toc282748829)

[60. Метод частной булевой производной синтеза тестов 43](#_Toc282748830)

[61. Цепной метод поиска булевой производной. 43](#_Toc282748831)

[62. Применение префиксной формы задания функции в цепном методе поиска булевой производной. 43](#_Toc282748832)

[63. Метод эквивалентных нормальных форм синтеза тестов. 43](#_Toc282748833)

[64. Синтез тестов для последовательностных схем. 44](#_Toc282748834)

[65.Контроль арифметических и логических операций. 44](#_Toc282748835)

[66.Требования, предъявляемые к тестам. 46](#_Toc282748836)

[67.Классификация методов тестового контроля. 46](#_Toc282748837)

[68.Методы сжатия реакций ОК. 47](#_Toc282748838)

[69. Функции счета. 47](#_Toc282748839)

[70. Контрольные суммы. 49](#_Toc282748840)

[71. Синдром. 50](#_Toc282748841)

[72. Спектральные коэффициенты 51](#_Toc282748842)

[73. Вероятностное тестирование 53](#_Toc282748843)

[74. Общая характеристика методов синтеза детерминированных тестов. 53](#_Toc282748844)

[Задачи 53](#_Toc282748845)

# Ответы на вопросы

## 1. Краткая характеристика процесса проектирования

Проектирование – разработка тех.документации на изготовление некоторого объекта с заданными свойствами или создание, преобразование и представление принятой формы образа еще не существующего объекта.

Требования – имеются ли физические средства для реализации, может уже кто-то придумал (технология), техническое задание. Проектирование включает комплекс работ исследовательского, расчетного и конструкторского характера.

Способы:

- ручной, автоматизированный, автоматический. Автоматизированный – комп выполняет большинство задач, больше или меньше в зависимости от стадии. Автоматических систем на данный момент не сущ.

## 2. Определение САПР

САПР (CAD) – организационно-техническая система, состоящая из комплекса средств автоматизированного проектирования, взаимосвязанных с определенными подразделениями проектных организаций или коллективом специалистов и выполняющих автоматизированное проектирование.

## 3. Виды обеспечения САПР

Выделяют следующие виды обеспечения САПР:

1. Техническое обеспечение (ТО) — совокупность связанных и взаимодействующих технических средств, обеспечивающих работу САПР, включающая различные аппаратные средства (ЭВМ,периферийные устройства, сетевое оборудование, линии связи, измерительные средства).

2. Математическое обеспечение (МО), объединяющее математические методы, модели и алгоритмы используемые для решения задач автоматизированного проектирования. МО по назначению и способам реализации делят на две части:

* математические методы и построенные на их основе математические модели объектов проектирования или их части;
* формализованное описание технологии автоматизированного проектирования.

3. Программное обеспечение (ПО), представляемое компьютерными программами необходимыми для осуществления процесса проектирования. ПО САПР подразделяется на общесистемное и прикладное:

* общесистемное ПО предназначено для управления компонентами технического обеспечения и обеспечения функционирования прикладных программ. Примером компонента общесистемного ПО является операционная система.
* прикладное ПО реализует математическое обеспечение для непосредственного выполнения проектных процедур, включает программы пакеты прикладных программ, предназначенные для обслуживания определенных этапов проектирования или групп однотипных задач внутри различных этапов (модуль проектирования трубопроводов, пакет схемотехнического моделирования, геометрический решатель САПР).

4. 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 и системы управления базами данных.

5. 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 ЛО выделяют класс различного типа языков проектирования и моделирования (VHDL, VERILOG, UML, GPSS).

6. Методическое обеспечение (МетО) — описание технологии функционирования САПР, методов выбора и применения пользователями технологических приемов для получения конкретных результатов, включающее в себя теорию процессов, происходящих в проектируемых объектах, методы анализа, синтеза систем и их составных частей, различные методики проектирования, иногда к МетО относят также МО и ЛО.

7. 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, должностные инструкции, 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 ее функционирования.

## 4. Принципы системного подхода к процессу проектирования

Система – набор примитивов, объектов и взаимосвязей для выполнения задачи. Основной принцип - рассмотрение систем по частям, обязательно с учетом их взаимодействий. Включает в себя выявление структуры системы, типизацию связей между ее элементами, определение атрибутов и анализ влияния внешней среды. Конкретизация выявляется другими названиями – блочно-иерархический, структурный, объектно-ориентированный. Структурный – синтезировать варианты системы из компонентов и оценить их при частичном переборе и предварительном прогнозировании х-к компонентов. Блочно-иерарх – декомпозиция сложных объектов и соотв средства их создания на иерархич уровни и аспекты вводит понятие стиля проектирования (восходящее и нисходящее). Объектно-ориент подход реализуется при проектировании ПО, разработке информац. Систем. Преимущ-ва: вносит в модели приложений большую структурную определенность, распределяя данные и процедуры между классами объектов. Сокращает объем спецификацию, благодаря введению в описание иерарх объектов. Уменьшает возможность искажения данных в следств ошибочных действий за счет ограничения доступа. Для всех подходов характерно след: структуризация процесса П, выраж декомпозицей проектных задач и документации, выделение стадий, этапов, процедур. Итерационный хар-р проектирования. Типизация и унификация проектных решений и средств П.

## 5. Общий подход к делению проектирования

Этапы проектирования:

1. Деление проектирования по времени вып. работы.

- НИР, на вых. Технические предложения на использование полученных результатов

- Опытно-конструкторские работы

- Рабочий проект. Разрабатывают конкретную тех.док-ю на изг. Пр. устр-ва. После этого изгот. опытный образец.

2. Вертикальное деление (по характеру учитываемых свойств объекта). 4 этапа:

- функционального проектирования (анализ ТЗ и корр., разработка структурных, функциональных, логич. и принципиальных схем пр. объекта)

- алгоритмическое пр-е (все что связано с ПО – от системы команд до ОС)

- конструкторское пр-е (документация для изгот. изделия – все что связано с физич. Реализацией)

- технологическое пр-е (вкл. Разработку док-ции на технологию изгот-ния проектируемого объекта)

3. Горизонтальное.

Каждый из этапов вертикального деления делятся, также по горизонтали:

ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

1. Системное проектирование; 2. Логическое проектирование; 3. Схемотехническое проектирование; 4. Компонентное проектирование.

АЛГОРИТМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

1. Программирование всей системы; 2. Программирование модулей; 3. Проектирование микропрограмм.

КОНСТРУКТОРСКОЕ ПРОЕКТИРОВАНИЕ

1. Проектирование "шкаф-стойка"; 2. Проектирование панелей; 3. Проектирование ТЭЗ (технический элемент замены); 4. Проектирование модуль, кристалл, ячейка.

ТЕХНОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

1. Разработка принципиальных схем технологического процесса; 2. Маршрутная технология; 3.Разработка технологических операций

## 6. Деления процесса проектирования по временному признаку

Научно-исследовательские работы(НИИ, НИ отделы). Оцениваются готовые проекты и предлагаются новые методы, компоненты, теории и тд. Результат – предложение испытаний результатов при проектировании новых изделий. Используются спец. САПР. Результаты исследований не являются обязательными для испытаний.

Опытно-конструкторские работы (ОКР) – формируется и согласовывается тех. Задание на разработку нового изделия и выполняются ОКР. Прорабатываются возможные вар-ты проекта, можно ли реализовать данное т/з, задаются элементная база, нет ли необходимости ее разработать. Синтезируется эских проекта. Результат – эскиз будущего устр-ва.

Техническое (рабочее) проектирование – основной этап П., на кот. Непосредственно разрабатывается проект. Рез-т – документация на изготовление П. (принц. Схемы со всеми описаниями)

Производство опытного образца

Испытания опытного образца. По рез-ам возможны вар-ты.

## 7. Деление процесса проектирования по характеру выполняемых работ

Функциональное. Исх данные – ТЗ и пром. рез-ты алгоритмического П. Цель – разработка функциональных, принц., структурных схем. Уточняется ТЗ, распределяются ф-ии объекта, кот будут выполняться при помощи технических средств.

Алгоритмическое – Разраб. Алгоритмы реализации ф-ий П, определяется о/с для работы, разраб. Программы и микропрограммы для объекта.

Конструкторское – разраб. конструкции(железо), кот. Будут реализовывать данный проект.

Технологическое – разр. технологии изготовления данного продукта, определяются маршрутные карты, инструменты для реализации и тд.

## 8. Деление процесса проектирования по блочно-иерархическому подходу

В основе лежит декомпозиция проекта по уровням П., выполняемых последовательно.

**Функциональное**

1. Системный уровень – определяется общая структура проекта; определяется функц. и структурная схема, после построения схемы определяется нужно ли проектировать новые эл-ты. Если да, то 2

2. Функц.-логический – опред. функц . и лог. эл-ты будущего устройства, в частности монтаж ; определяется заполн. платы реальными эл-ами; компоновка и размещение эл-ов в модулях; трассировка печатной платы, анализируется электрич. парам. Будущего изделия. Если необх. выполн. П компонентов. Передается документация на уровень П.

3. Технический

4. Компонентный

**Алгоритмическое**

1. П. алгоритмов.

2. Программирование системы

3. Программ. Модулей

4. П. мета-программ. – программисты

**Конструкторское**

Шкаф, стойка, панель, модуль, кристалл ячейка – конструкторы

**Техническое** – принц. схема, технолог. процесса, маршрутные карты, технолог. операции – технологи

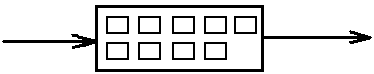
## 9. Блочно-иерархический подход к процессу проектирования.

Используется декомпозиция описания сложных объектов и соотв средств для их создания на иерархические уровни и аспекты, вводит понятие списка проектирования(восходящее и нисходящее)

На верхнем этапе устройство представляется как черный ящик.

Техническое задание

Системный



Определяются какие ф-ии в каких эл-ах будут выполняться.

Затем определяется, какие эл-ты нужно разработать.

С каждым этапом все более высокий уровень детализации.

«Линии отрыва» между уровнями определяются матем. аппаратом, кот может быть использован при проект. на данном этапе.

1) На самом верхнем этап исп.теория вычислительных систем для решения задачи синтеза и имитационное моделирование для анализа.

2) Синтез – ПТЦА (прикладная теория цифровых автоматов). Анализ – логическое моделирование. Получаем функц. и логические схемы.

3) Синтез и анализ – теория графов, теория множеств.

## 10. Типовые проектные процедуры.

Основные – синтез и анализ. Синтез – непосредственная разработка проекта. Анализ – проверка правильности полученных рез-ов. Сущность П заключается в принятии проектных решений, обеспечивающих выполнение предъявленных требований. Синтез – основа. П. Анализ – вспомогательная но необх. процедура.

Синтез:

\* структурный – разработка структуры будущего объекта

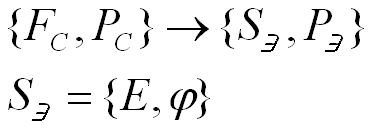
\* параметрический – определение параметров, реализуемых данным объектом.

Задача параметрического синтеза является задачей оптимизации, если целью является улучшения х-к объекта, либо выбором синтеза наилучшего решения данной задачи.

Постановка задачи структурно синтеза

O = { F, S, P }, O-описание, F-функц, S – структ, P- параметрические.

Структурный синтез – преобразование функц. описания и параметрического описания сис-мы в структуру и парам описание эл-ов.



## 11. Задача синтеза в процессе проектирования.

Синтез – создание описания объекта, выполняющего заданные функции и удовлетворяющего заданным ограничениям.

Описание – набор инструкций в каком – либо алфавите.

Задача синтеза выполняется в выбранном классе элементарных объектов, из кот. составляется объект, реализующий заданный класс функций.

Исх. данные: описание ф-ций, возлагаемых на пректируемый объект; перечень параметров, характериз. качество и ограничения на их значения.

Результат – некоторая структура, реализующая заданный класс ф-ций.

Под СТРУКТУРОЙ объекта понимается множество S = {C,H}, где С – множество элементов, входящих в структуру объекта, а H - множество связей между ними.

Две структуры называются равными, если они реализуют равные функции

(F1 = F2), состоит из одинаковых элементов ({C1} = {C2}), кот. связаны одинаковыми связями ({H1} = {H2}).

Две структуры называются ЭКВИВАЛЕНТНЫМИ, если F1 = F2, но

C1<>C2 и(или) H1<>H2.

Задача синтеза может иметь формальные методы решения – такая задача алгоритмически разрешима, иначе алгоритмически неразрешима. Алгоритмически – неразрешимые задачи решаются в ручную или с помощью эвристических методов (полный перебор).

Различают СИНТЕЗ СТРУКТУРНЫЙ и СИНТЕЗ ПАРАМЕТРИЧЕСКИЙ. Цель структурного синтеза - получение структурных схем объекта, содержащих сведения о составе элементов и способах соединения их между собой. Цель параметрического синтеза – определение числовых значений параметров элементов.

Синтез называется ОПТИМИЗАЦИЕЙ, если определяются наилучшие, в заданном смысле, структуры и значение параметров. Задачу выбора оптимальной структуры называют СТРУКТУРНОЙ ОПТИМИЗАЦИЕЙ.

При расчете оптимальных значений параметров при заданной структуре говорят о ПАРАМЕТРИЧЕСКОЙ ОПТИМИЗАЦИИ.

## 12. Задача анализа в процессе проектирования.

АНАЛИЗ - это определение функционального и параметрического описания системы по заданному структурному описанию.

Предмет решения задачи анализа – исследование свойств F, S и Р-описаний, полученных на некотором шаге при спуске по дереву проектных решений. Целью такого исследования является оценка качества полученного варианта решения или верификация F-описания на соответствие заданному.

В отличие от задачи синтеза, задача анализа алгоритмически всегда разрешима. Утверждение справедливо, поскольку вариант решения задачи синтеза уже получен и известны, по крайней мере, соответствующие ему F и S – описания.

Задача анализа решается с помощью моделирования.

Наиболее общими методами анализа являются одновариантный (исследование объекта в заданной точке траектории поведения) и многовариантный (исследование свойств объекта в окресностях заданной точки траектории поведения).

Адекватность – показатель соответствия модели анализируемому объекту.

## 13. Задача оптимизации в процессе проектирования.

Обобщенная постановка задачи оптимизации

Оптимальными считаются те значения, которые удовлетворяют ТЗ и являются лучшими из достижимых.

Задача оптимизации САПР сводится к преобразованию физического представления об объекте, о его назначении и степени полезности в математическую формулировку экстремальной задачи. Цель оптимизации выражается в критериях оптимизации.

Критерии – правила предпочтения сравниваемых вариантов. Основу критериев оптимизации составляет целевая ф – ция F(x), где х – мн-во управляемых параметров. Векторы х с фиксированными значениями определяют один из вариантов объекта и его характеристики.

Целевая функция должна быть такой, чтобы по ее значениям можно было определить степень достижения цели, т.е. лучший вариант должен характеризоваться большим значением F(X), тогда оптимизация заключается в максимализации F(X) или наоборот при минимизации F(X) лучший вариант должен характеризоваться меньшими значениями параметров.

Кроме целевой функции F(X) и перечня управляемых параметров (X) в постановку задачи оптимизации могут входить ОГРАНИЧЕНИЯ ТИПА РАВЕНСТВ H(X) = 0 и НЕРАВЕНСТВ H(X)<>0. Частным случаем ограничений типа неравенств являются прямые ограничения ai<=xi<=bi, где ai и bi - предельно допустимые значения параметра xi.

Область пространства управляемых параметров, в которой выполняются заданные ограничения, называет ДОПУСТИМОЙ ОБЛАСТЬЮ XD.

Объект называется строго оптимальным, если значения всех параметров находятся в допустимой области значений параметров.

Объект называется квазиоптимальным, если некоторые параметры из вектора (Х) выходят за границы ограничений, но при этом ограничения границ должны быть строго заданы.

При наличии ограничений задача оптимизации называется УСЛОВНОЙ ОПТИМИЗАЦИИ, в противном случае – БЕЗУСЛОВНОЙ.

Область, в которой выполняются как прямые ограничения, так и условия работоспособности, называется ОБЛАСТЬЮ РАБОТОСПОСОБНОСТИ.

Таким образом, итоговая формулировка задачи оптимизации при проектировании имеет вид: экстремизировать целевую функцию F(X) в области XD, заданной ограничениями H(X) = 0 и ф(X) > 0.

Задача оптимизации в такой постановке есть задача математического программирования. При линейности функций F(X), H(X), ф(X) - задача линейного программирования. Если хотя бы одна из них нелинейна - задача нелинейного программирования..

Если все (или часть) X - дискретны, то задача дискретного (или частично дискретного) программирования.

Дискретное программирование называется целочисленным, если X принадлежит множеству целых чисел. Если XD есть пространство булевых переменных, то – задача бивалентного программирования.

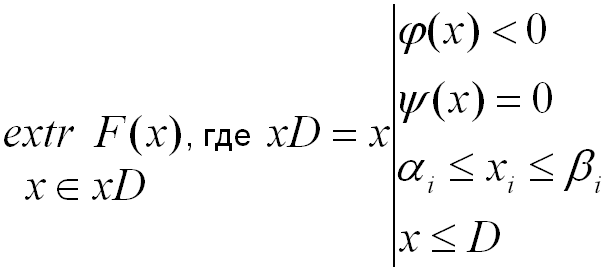
Задача структурной оптимизации сводится к построению оптимальной структуры S = (E,H). При этом под ОПТИМАЛЬНЫМ будем понимать такой вариант структуры, параметры которой удовлетворяют всем системным, конструктивным, технологическим, электрическим и экономическим требованиям ТЗ, а критерий оптимальности, описывающий качество проектируемой структуры, принимает экстремальное значение.

## 14. Математическая постановка задачи оптимизации.

Оптимальный – удовлетворяющий целевую ф-ию и укладывающийся в имеющиеся ресурсы.

Постановка задачи: следует преобразовать (блаблабла) в математическую формулировку экстремальной задачи. Цель оптимизации выражается в критериях оптимизации. Основа критерия – целевая ф-я F(x), где х – множество управляющих параметров. Фиксация знчаий вектора управляемых параметров представляет некоторое решение задачи оптимизации. Как правило, в вектор входят все параметры, которые характеризуют объект, либо часть их, тогда остальные либо фиксированы, либо заданы областями. Следовательно, на часть параметров накладывается некоторые ограничения. Ограничения задаются математически в виде неравенств либо равенств, либо прямые ограничения. Задача оптимизации наз задачей условной оптимизации, если все ограничения заданы неравенствами. Безусловной – равенствами. Область параметров, кот. удовлетворяют области ограничений, наз. допустимой областью значений. XD

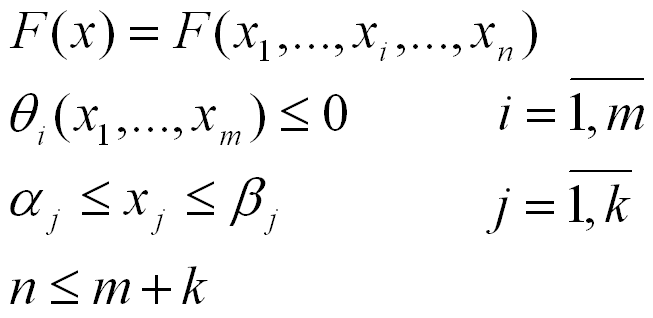
В области допустимых значений параметров экстремум f(x)



Задача оптимизации в такой постановке – это задача мат. программирования. Если все ф-ии линейны – задача линейна. Если хоть одна нелинейна – задача нелинейна.

Если все или часть параметров дискретны – соотв дискретны xeZ, или частично дискретны – целочисленное прогр. xe{0,1} – бивалентное программирование. При структурной оптимизации необходимо найти оптимальный вар-т – множество эл-ов и связей между ними.

Оптимальная – структура, параметры которой удовлетворяют всем системным, конструктивным, технологическим и эконом. требованиям ТЗ, а критерий оптимальности принимает экстремальное значение. Следовательно в формализованном виде задача структ. оптимизации заключается в определении значений независимых переменных X­I при которой критерий оптимальности F(x) есть множество независимых переменных принимается экстремальное значение.



## 15. Общая характеристика критериев оптимизации.

Критерий оптимальности – это правильно предпочтений сравниваемых вар-ов. Критерий который характеризует весь объект называется частным критерием. Каждый критерий можно назвать частным, если он не взаимодействует со всеми остальными.

Если в функции можно задать хар-ки всех необходимых критериев, такой критерий называется обобщенным. В качестве обобщенных критериев наиболее часто используется аддитивный, мультипликативный, минимаксный. Если критерий не учитывает вероятностный разброс параметров – он детерминированный, иначе – статистический.

В аддитивных критериях целевая функция образуется путем сложения нормированных значений частных критериев. Нормированные критерии представляют собой отношение реального значения частного критерия к некоторой нормирующей величине, измеряемой в тех же единицах, что и сам критерий (приводит к безразмерной величине).

Возможны несколько подходов к выбору нормирующего делителя.

Первый подход предполагает принимать в качестве нормирующих делителей максимальных значений критериев, достигаемых в области существующих проектных решений.

Второй подход предполагает принимать в качестве нормирующих делителей то оптимальное значение, кот. задано в ТЗ.

Третий подход предполагает в качестве нормирующих делителей использовать разность между максимальным и минимальным значением критерия в области компромисса.

Целевая ф-ция: F(x) = SCi\*fi(x) =SCi \*Fi(x)/Fнорм.i(x)

Недостатки: формальный прием, не вытекает из объективной роли часного критерия; может происходить взаимная компенсация частных критериев.

Мультипликативный критерий. Иногда, важно учитывать не абсолютное значение критерия, а его изменение при решении некоторой задачи.

Целевая ф-ция: F(x) = П Fi(x)

В случае неравноценности частных критериев необходимо ввести весовой коэффициент Ci и тогда мультипликативный критерий примет вид:

n Ci n

F(x) = П Fi (x) или F(x) = П Ci Fi (x)

i=1 i=1

Достоинством мультипликативного критерия является то, что при его использовании не требуется нормирования частных критериев.

Недостаток: критерий может компенсировать чрезмерные изменения одних критериев за счет изменения других.

Минимаксный критерий.

Формально принцип максмина формулируется следующим образом:

Необходимо выбирать такое множество X0 е X, на котором реализуется максимум из минимальных значений частных критериев F(x0) = max min { fi(x) }.

Если частные критерии fi(x) следует минимизировать, то используется принцип минимакса

F(x0) = min max {fi(x)}.

Аддитивные критерии выбирают, когда существенное значение имеют абсолютные числовые значения критериев при выбранном векторе X.

## 16. Аддитивный критерий оптимизации. = 15

В аддитивных критериях целевая функция образуется путем сложения нормированных значений частных критериев. Нормированные критерии представляют собой отношение реального значения частного критерия к некоторой нормирующей величине, измеряемой в тех же единицах, что и сам критерий (приводит к безразмерной величине).

Возможны несколько подходов к выбору нормирующего делителя.

**Первый подход** предполагает принимать в качестве нормирующих делителей максимальные значения критериев, достигаемые в области существующих проектных решений(в мировой практике).

**Второй подход** предполагает принимать в качестве нормирующих делителей то оптимальное(наилучшее) значение, которое задано в ТЗ.

Третий подход предполагает в качестве нормирующих делителей использовать разность между максимальным и минимальным значением критерия в области компромисса.

Целевая ф-ция: F(x) = SCi\*fi(x) =SCi \*Fi(x)/Fнорм.i(x), где Ci – коэффициент значимости i-го критерия(чаще всего находят методом экспертных оценок), Fнорм.i(x) – нормированная величина

Недостатки:

- формальный прием, не вытекает из объективной роли частного критерия;

- может происходить взаимная компенсация частных критериев.

## 17. Мультипликативный критерий оптимизации.

Иногда, важно учитывать не абсолютное значение критерия, а его изменение при решении некоторой задачи.

Целевая ф-ция: F(x) = П Fi(x)

В случае неравноценности частных критериев необходимо ввести весовой коэффициент Ci и тогда мультипликативный критерий примет вид:

n Ci n

F(x) = П Fi (x) или F(x) = П Ci Fi (x)

i=1 i=1

Достоинством мультипликативного критерия является то, что при его использовании не требуется нормирования частных критериев.

Недостаток: критерий может компенсировать чрезмерные изменения одних критериев за счет изменения других.

## 18. Минимаксные критерии оптимизации.

Формально принцип максмина формулируется следующим образом:

Необходимо выбирать такое множество X0 е X, на котором реализуется максимум из минимальных значений частных критериев F(x0) = max min { fi(x) }.

Если частные критерии fi(x) следует минимизировать, то используется принцип минимакса

F(x0) = min max {fi(x)}.

## 19. Частные критерии оптимизации.

При проектировании по частным критериям в качестве целевой функции F(X) применяется наиболее важный выходной параметр проектируемого объекта, все остальные параметры в виде соответствующих условий работоспособности относятся к ограничениям. В этом случае задача оптимального проектирования является однокритериальной задачей математического программирования: экстремизировать значение целевой функции F(X) при наличии системы ограничений на параметры проектируемого объекта. Сложность такой задачи небольшая.

Частные критерии выбирают тогда, когда необходимо сравнить несколько эквивалентных решений, либо заранее задана необходимость оптимизации одного или нескольких частных критериев (без существенных ограничений на другие критерии).

## 20. Методы задания предпочтений на множестве частных критериев в задаче оптимизации.

Если можно выделить 1 параметр, который наиболее полно описывает объект, то оптимизация ведется по частному признаку, а на остальные накладываются ограничения.

Если все ограничения заданы в виде равенств – задача алгебраическая.

Если все значения параметров заданы в виде равенств и неравенств – задача не имеет степеней свободы.

Частные критерии имеют ряд недостатков: Достаточно часто при выборе экстремального значения 1-го частного критерия, остальные, как правило, даже при наличии ограничений принимают неудовлетворительные значения.

Пример:

|  |  |
| --- | --- |
| Пусть необходимо оптимизировать по y1 и y2  y1 <= T1  y2 <= T2  Экстремальное значение параметров будет в точках A и B. Кривая указывает на те точки области работоспособности, которые имеют наилучшие значение параметров.  Множество точек пространства, из которых невозможно перемещение, приводящих к улучшению всех параметров, называется областью Парета.Частный критерий у1 дает как у2. Но тогда при нестабильности работы системы он будет «гулять» в т.ч. вне области.Тогда берут разницу (∆) у1 <= T1 + ∆, но ∆ выбрать сложно.  Поэтому целесообразно брать обобщенный критерий. |  |

## 21. Общая характеристика системного уровня проектирования.

Системный уровень – определяется общая структура проекта; определяется функц. и структурная схема, после построения схемы определяется нужно ли проектировать новые эл-ты. Если да, то переходим на функц.-логический этап проектирования.

Функц.-логический – опред. функц . и лог. эл-ты будущего устройства, в частности монтаж ; определяется заполн. платы реальными эл-ами; компоновка и размещение эл-ов в модулях; трассировка печатной платы, анализируется электрич. парам. Будущего изделия. Если необх. выполн. П компонентов. Передается документация на уровень П.

## 22. Задачи синтеза и анализа на системном уровне проектирования.(мб это не то)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вертикальные уровни | Горизонтальные уровни |  |  |  |
| Функциональный | — | Системный  Структурный  (структурные схемы) | Функционально-логический  (функ­циональные и принципиальные схемы) | Схемо­технический  (принципиальные электрические схемы ИС) |
| Алгоритмический | Проектирование алгоритмов и программирование  (ПО ЭВМ, ОС ЭВМ) | Архитектурный  (система ко­манд, форматы данных команд, системное об­служивание пре­рываний) | Микропрограммный  (проектирование микропрограмм, операций и про­цедур, вы­пол­няе­мых аппаратным способом) | — |
| Конструкторский | Сети, системы | Блок, комплекс, система | Плата, ТЭЗ, модуль | Кристалл, микросхема |

Теперь рассмотрим основные задачи, решаемые на различных уровнях.

При нисходящем проектировании связь иерархических уровней проявляется через формирование ТЗ на разработку элементов исходя из требований, предъявляемых к системе. Формирование ТЗ на некоторой объект производится на более высоком уровне по отношению к уровню данного объекта.

Системный уровень. Архитектурный уровень

Для **системного** и **архитектурного** уровней эти задачи следующие:

– определение принципов организации ВС;

– выбор архитектуры, уточнение функций ВС и их разделение на функции, реализуемые аппаратным и программным путем;

– разработка структурных схем, определения состава устройств и способов их взаимодействия;

– определение требований к выходным параметрам устройств и формирование ТЗ на разработки отдельных устройств ЭВМ.

## 23. Общая характеристика СМО.

**Система массового обслуживания** (СМО) - система, которая производит обслуживание множество заявок (система с очередями). Состоит из:

* стационарных объектов
* обслуживающих аппаратов
* хранилищ

Типичными выходными параметрами в СМО являются числовые характеристики таких величин, как время обслуживания заявок в системе, длины очередей заявок на входах, время ожидания обслуживания в очередях, загрузка устройств системы, а также вероятность обслуживания в заданные сроки и т. п.

В простейшем случае СМО представляет собой некоторое средство (устройство), называемое обслуживающим аппаратом (OA), вместе с очередями заявок на входах. Более сложные СМО состоят из многих взаимосвязанных OA.

Обслуживающие аппараты СМО в совокупности образуют статические объекты СМО, иначе называемые ресурсами. Например, в вычислительных сетях ресурсы представлены аппаратными и программными средствами.

В СМО кроме статических объектов фигурируют динамические объекты — транзакты. Например, в вычислительных сетях динамическими объектами являются решаемые задачи и запросы на информационные услуги.

Состояние СМО характеризуется состояниями составляющих ее объектов. Например, состояния OA выражаются булевыми величинами, значения которых интерпретируются как true (занято) и false (свободно), и длинами очередей на входах OA, принимающими неотрицательные целочисленные значения. Переменные, характеризующие состояние СМО, будем называть переменными состояния или фазовыми переменными.

Правило, согласно которому заявки выбирают из очередей на обслуживание, называют дисциплиной обслуживания, а величину, выражающую преимущественное право на обслуживание, — приоритетом. В бесприоритетных дисциплинах все транзакты имеют одинаковые приоритеты. Среди бесприоритетных дисциплин наиболее популярны дисциплины FIFO (первым пришел — первым обслужен), LIFO (последним пришел — первым обслужен) и со случайным выбором заявок из очередей.

В приоритетных дисциплинах для заявок каждого приоритета на входе ОА выделяется своя очередь. Заявка из очереди с низким приоритетом поступает на обслуживание, если пусты очереди с более высокими приоритетами. Различают приоритеты абсолютные, относительные и динамические. Заявка из очереди с более высоким абсолютным приоритетом, поступая на вход занятого ОА, прерывает уже начатое обслуживание заявки более низкого приоритета. В случае относительного приоритета прерывания не происходит, более высокоприоритетная заявка ждет окончания уже начатого обслуживания. Динамические приоритеты могут изменяться во время нахождения заявки в СМО.

Исследование поведения СМО, т. е. определение временных зависимостей переменных, характеризующих состояние СМО, при подаче на входы любых требуемых в соответствии с заданием на эксперимент потоков заявок, называют имитационным моделированием СМО. Имитационное моделирование проводят путем воспроизведения событий, происходящих одновременно или последовательно в модельном времени. При этом под событием понимают факт изменения значения любой фазовой переменной.

Подход, альтернативный имитационному моделированию, называют аналитическим исследованием СМО. Аналитическое исследование заключается в получении формул для расчета выходных параметров СМО с последующей подстановкой значений аргументов в эти формулы в каждом отдельном эксперименте.

Модели СМО, используемые при имитационном и аналитическом моделировании, называются имитационными и аналитическими соответственно.

Основным подходом к анализу САПР на системном уровне проектирования считают имитационное моделирование, а аналитическое исследование используют при предварительной оценке различных предлагаемых вариантов систем.

## 24. Аналитическая модель СМО.

Аналитические модели удобны в использовании, поскольку для аналитического моделирования не требуются сколько-нибудь значительные затраты вычислительных ресурсов, часто без постановки специальных вычислительных экспериментов разработчик может оценить характер влияния аргументов на выходные параметры, выявить те или иные общие закономерности в поведении системы. Но, к сожалению, аналитическое исследование удается реализовать только для частных случаев сравнительно несложных СМО. Для сложных СМО аналитические модели если и удается получить, то только при принятии упрощающих допущений, ставящих под сомнение адекватность модели. Аналитическое исследование используют при предварительной оценке различных предлагаемых вариантов систем.

Аналитические модели СМО удается получить при довольно серьезных допущениях. К числу типичных допущений относятся следующие:

* Во-первых, как правило, считают, что в СМО используются бесприоритетные дисциплины обслуживания типа FIFO.
* Во-вторых, времена обслуживания заявок в устройствах выбираются в соответствии с экспоненциальным законом распределения.
* В-третьих, в аналитических моделях СМО входные потоки заявок аппроксимируются простейшими потоками, т. е. потоками, обладающими свойствами стационарности, ординарности (невозможности одновременного поступления двух заявок на вход СМО), отсутствия последействия.

В большинстве случаев модели СМО отображают процессы с конечным множеством состояний и с отсутствием последействия. Такие процессы называют **конечными марковскими цепями**.

Марковские цепи характеризуются множеством состояний S, матрицей вероятностей переходов из одного состояния в другое и начальными условиями (начальным состоянием). Удобно представлять марковскую цепь в виде графа, в котором вершины соответствуют состояниям цепи, дуги — переходам, веса дуг — вероятностям переходов (если время дискретно) или интенсивностям переходов (если время непрерывно).

Отметим, что интенсивностью перехода называют величину Vij = lim (Pij(t) / t) при t -> 0, где - Pij(t) - вероятность перехода из состояния Si в состояние Sj за время t. Обычно используют условие

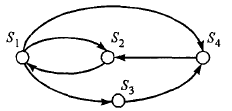


что эквивалентно



где N — число состояний.

Приведем пример марковской цепи:



Матрица интенсивностей для этого примера:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Состояние | S1 | S2 | S3 | S4 |
| S1 | -V12 - V13 - V14 | V12 | V13 | V14 |
| S2 | V21 | -V21 | 0 | 0 |
| S3 | 0 | 0 | -V34 | V34 |
| S4 | 0 | V42 | 0 | -V42 |

Большинство выходных параметров СМО можно определить, используя информацию о поведении СМО, т. е. информацию о состояниях СМО в установившихся (стационарных) режимах и об их изменениях в переходных процессах. Эта информация имеет вероятностную природу, что обусловливает описание поведения СМО в терминах вероятностей нахождения системы в различных состояниях. Основой такого описания, а следовательно, и многих аналитических моделей СМО являются уравнения Колмогорова.

Уравнения Колмогорова можно получить следующим образом.

Изменение вероятности Р нахождения системы в состоянии Si за время t есть вероятность перехода системы в состояние Si из любых других состояний за вычетом вероятности перехода из состояния Si в другие состояния за время t, т. е.



Вероятность равна разнице:





## 25. Имитационная модель СМО.

Основным подходом к анализу САПР на системном уровне проектирования считают **имитационное моделирование**. Имитационное моделирование проводят путем воспроизведения событий, происходящих одновременно или последовательно в модельном времени. При этом под событием понимают факт изменения значения любой фазовой переменной.

Для представления имитационных моделей можно использовать языки программирования общего применения, однако такие представления оказываются довольно громоздкими. Поэтому обычно используют специальные языки имитационного моделирования на системном уровне. Среди языков имитационного моделирования различают языки, ориентированные на описание событий, средств обслуживания или маршрутов движения заявок (процессов). Выбор языка моделирования определяет структуру модели и методику ее построения.

Ориентация на устройства характерна для функционально-логического и более детальных иерархических уровней описания объектов.

Для описания имитационных моделей на системном уровне (иногда их называют сетевыми имитационными моделями — СИМ) чаще используют языки, ориентированные на события или процессы. Примерами первых могут служить языки Симскрипт, SMPL и ряд других. К числу вторых относятся языки Симула, SOL, а также популярный язык GPSS.

Языки имитационного моделирования реализуются в программно-методических комплексах моделирования СМО, имеющих ту или иную степень специализации. Так, комплексы на базе языка GPSS можно использовать во многих приложениях, но есть специализированные комплексы для моделирования вычислительных сетей, систем управления предприятиями и т. п.

При использовании языков, ориентированных на процессы, в составе СИМ выделяются элементарные части и ими могут быть источники входных потоков заявок, устройства, накопители и узлы.

Источник входного потока заявок представляет собой алгоритм, в соответствии с которым вычисляются моменты tk появления заявок на выходе источника. Источники могут быть зависимыми и независимыми. В зависимых источниках моменты появления заявок связаны с наступлением определенных событий, например с приходом другой заявки на вход некоторого устройства. Типичным независимым источником является алгоритм выработки значений случайной величины с заданным законом распределения.

Устройства в имитационной модели представлены алгоритмами выработки значений интервалов (длительностей) обслуживания. Чаще всего это алгоритмы генерации значений случайных величин с заданным законом распределения. Но могут быть устройства с детерминированным временем обслуживания или временем, определяемым событиями в других частях СИМ. Модель устройства отображает также заданную дисциплину обслуживания, поскольку в модель входит алгоритм, управляющий очередями на входах устройства.

Накопители моделируются алгоритмами определения объемов памяти, занимаемых заявками, приходящими на вход накопителя. Обычно объем памяти, занимаемый заявкой, вычисляется как значение случайной величины, закон и (или) числовые характеристики распределения могут зависеть от типа заявки.

Узлы выполняют связующие, управляющие и вспомогательные функции в имитационной модели, например, для выбора направлений движения заявок в СИМ, изменения их параметров и приоритета, разделения заявок на части, их объединения и т. п.

Обычно каждому типу элементарной модели, за исключением лишь некоторых узлов, в программной системе соответствует определенная процедура (подпрограмма). Тогда СИМ можно представить как алгоритм, состоящий из упорядоченных обращений к этим процедурам, отражающим поведение моделируемой системы.

В процессе моделирования происходят изменения модельного времени, которое чаще всего принимается дискретным, измеряемым в тактах. Время изменяется после того, как закончена имитация очередной группы событий, относящихся к текущему моменту времени Имитация сопровождается накоплением в отдельном файле статистики таких данных, как количества заявок, вышедших из системы обслуженными и необслуженными, суммарное время занятого состояния для каждого из устройств, средние длины очередей и т. п. Имитация заканчивается, когда текущее время превысит заданный отрезок времени или когда входные источники выработают заданное число заявок. После этого производят обработку накопленных в файле статистики данных, что позволяет получить значения требуемых выходных параметров.

## 26. Динамическая структура моделирования СМО.

## 27. Событийный метод моделирования СМО.

В программах имитационного моделирования СМО преимущественно реализуется событийный метод организации вычислений. Сущность событийного метода заключается в отслеживании на модели последовательности событий в том же порядке, в каком они происходили бы в реальной системе. Вычисления выполняют только для тех моментов времени и тех частей (процедур) модели, к которым относятся совершаемые события. Другими словами, обращения на очередном такте моделируемого времени осуществляются только к моделям тех элементов (устройств, накопителей), на входах которых в этом такте произошли изменения. Поскольку изменения состояний в каждом такте обычно наблюдаются лишь у малой доли ОА, событийный метод может существенно ускорить моделирование по сравнению с инкрементным методом, в котором на каждом такте анализируются состояния всех элементов модели.

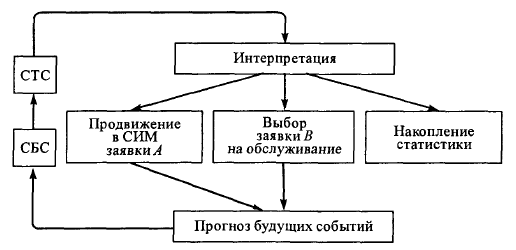
Рассмотрим возможную схему реализации событийного метода имитационного моделирования.

Моделирование начинается с просмотра операторов генерирования заявок, т. е. с обращения к моделям источников входных потоков. Для каждого независимого источника такое обращение позволяет рассчитать момент генерации первой заявки. Этот момент вместе с именем — ссылкой на заявку — заносится в список будущих событий (СБС), а сведения о генерируемой заявке — в список заявок (СЗ). Запись в СЗ включает в себя имя заявки, значения ее параметров (атрибутов), место, занимаемое в данный момент в СИМ. События в СБС упорядочиваются по мере увеличения моментов наступления.

Затем из СБС выбирают совокупность сведений о событиях, относящихся к наиболее раннему моменту времени. Эта совокупность переносится в список текущих событий (СТС), из которого извлекаются ссылки на события. Обращение по ссылке к СЗ позволяет установить место в СИМ заявки А, с которой связано моделируемое событие. Пусть этим местом является устройство X. Далее программа моделирования выполняет следующие действия:

1. изменяет параметры состояния устройства X; например, если заявка А освобождает X, а очередь к X не была пуста, то в соответствии с заданной дисциплиной обслуживания из очереди к X выбирается заявка В и поступает на обслуживание в X;
2. прогнозируется время наступления следующего события, связанного с заявкой В, путем обращения к модели устройства X, в которой рассчитывается продолжительность обслуживания заявки В; сведения об этом будущем событии заносятся в СБС и СЗ;
3. происходит имитация движения заявки А в СИМ по маршруту, определяемому заданной программой моделирования, до тех пор, пока заявка не придет на вход некоторого OA; здесь либо заявка задерживается в очереди, либо путем обращения к модели этого OA прогнозируется наступление некоторого будущего события, связанного с дальнейшей судьбой заявки А; сведения об этом будущем событии также заносятся в СБС и СЗ;
4. в файл статистики добавляются необходимые данные.

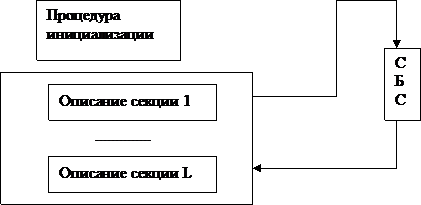
После отработки всех событий, относящихся к моменту времени tk, происходит увеличение модельного времени до значения, соответствующего ближайшему будущему событию, и рассмотренный процесс имитации повторяется.



## 28. Языки моделирования, ориентированные на описание событий.

Моделирующая программа организованна в совокупности в виде секций событий (процедуры отражающие события). Процедура состоит из набора операций, которые выполняются после выполнения какой-либо активности. Синхронизация происходит с помощью списка будущих событий.

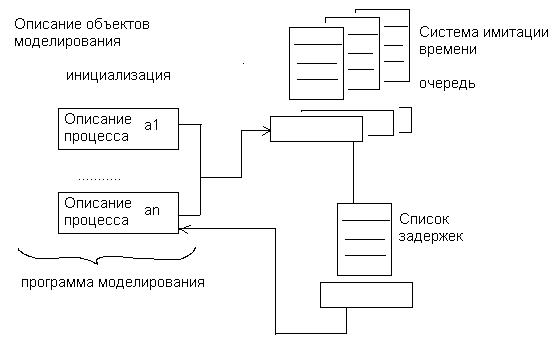
Структуру [рассмотрим](http://www.google.com/url?q=http%3A%2F%2Fwww.adpcom.ru%2Fnout%2Ffaq_ibm.shtml&sa=D&sntz=1&usg=AFQjCNFG-9qF2Dev0aR9s1oGqTybMYsD7w) на примере языка SIMSCRIPT.



## 29. Языки моделирования, ориентированные на описание процессов.

Моделирующая программа организуется в виде набора описаний процесса. Каждый из которых описывает одни класс. Описание процесса функционирования устанавливает атрибуты и активности всех процессов. Синхронизация операций во времени реализуются так же с помощью списка будущих событий, который содержит точку возобновления конкретного процесса (точка прерывания).

На примере языка SIMULA:



## 30. Краткое описание ASPOL.

ASPOL - язык, ориентированный на описание процессов.

Основная программа описывается следующим образом:

Sim <имя>; основная программа – описание программы, переменных, памяти, инициализация. Главный процесс в системе.

End.

Описание процесса выглядит следующим образом:

Process <имя>(p1,p2,p3...pn); формальные параметры

End process <имя>;

Инициализация экземпляра процесса выполняется:

Initiate <имя процесса>(a1,a2,a3,...) фактические параметры

Любой процесс может находится в одном из 4х состояний:

1. Готовности – процесс уже инициирован, но ещё не выполняется.
2. Выполнения
3. Задержки (hold (t))
4. Ожидания (ожидание освобождения устройства)

Процесс может вызываться из другого процесса. С каждым процессом связан приоритет, при инициализации он получает приоритет процесса-инициатора. Приоритет может быть изменен:

priority <выражение>;

С выполнением процесса связаны события, в ASPOL события могут быть заданы явно и неявно. События следования, как правило, задаются неявно. События изменения состояния могут быть заданы явно.

Event e1,e2,e3...en; объявление событий

Set(e1); событие сработало

Queue(e); если есть очередь событий

**Устройства:**

Facility <имя устройства>

В момент создания устройство не занято.

Reserve <имя> занять устройство

Release <имя> освободить устройство

Reserve(f,i) Если множество есть одинаковых устройств и надо занять/освободить i-ое

Release(f,i)

**Память:**

Storage(<имя памяти>) указывает, какие памяти есть

Size(s) <выражение> указывает весь объем памяти s

Allocate(s) <выражение> сколько памяти s надо занять

Deallocate(s) <выражение> сколько памяти s надо освободить

**Распределения:**

expute(U) экспоненциальный закон с мат. ожиданием U

erlang(U,V) U – мат. Ожидание, V - дисперсия

random(a,b)

pr = random(0,1)

irandom(a,b)

## 31. Краткое описание GPRS (GPSS).

GPSS ([англ.](http://www.google.com/url?q=http%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%25D0%2590%25D0%25BD%25D0%25B3%25D0%25BB%25D0%25B8%25D0%25B9%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA&sa=D&sntz=1&usg=AFQjCNGw8gO-iF9-AalJTVC4duOZnaX71w) General Purpose Simulation System — общецелевая система моделирования) — [язык программирования](http://www.google.com/url?q=http%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%25D0%25AF%25D0%25B7%25D1%258B%25D0%25BA_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D1%258F&sa=D&sntz=1&usg=AFQjCNHy4DyE_ONsSxGSgjBopAcXzMNjHw), используемый для имитационного [моделирования](http://www.google.com/url?q=http%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%25D0%259C%25D0%25BE%25D0%25B4%25D0%25B5%25D0%25BB%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5&sa=D&sntz=1&usg=AFQjCNH94F1ZoKbtOKBL5lDHKr4HM4GJbA)различных систем, в основном систем массового обслуживания.

Сейчас GPSS используется, но редко, так как менее гибок чем такие языки моделирования как [Simula](http://www.google.com/url?q=http%3A%2F%2Fru.wikipedia.org%2Fwiki%2FSimula&sa=D&sntz=1&usg=AFQjCNEf8-HH3TaxBA3wYYTqJNc34nKJxw) и [SIMSCRIPT II.5](http://www.google.com/url?q=http%3A%2F%2Fru.wikipedia.org%2Fw%2Findex.php%3Ftitle%3DSIMSCRIPT_II.5%26action%3Dedit%26redlink%3D1&sa=D&sntz=1&usg=AFQjCNFMzrXRai0VQVDybIt_RIexEv5Yjg).

Программа на языке GPSS состоит из блоков, которые имитируют различные параметры «устройств» в модели. Как то ожидание, выполнение работы и другие. Блок начинает выполняться при попадании в него транзактов — активных, неделимых элементов модели. К транзактам применяются различные правила описанные в блоках программы.

Пример кода для системы GPSS World

SIMULATE  
GENERATE (EXPONENTIAL(1,0.014,0.118)) ; генерация потока транзактов  
 ; с параметрами экспоненциального распределения   
ADVANCE (Normal(1,0.091,0.016)) ; Ожидание транзактом в течение времени,  
 ; распределённому по нормальному (гауссову) закону  
TERMINATE ; удаление транзакта

## 32. Пример аналитической модели СМО.

Аналитической моделью СМО является совокупность уравнений или формул, позволяющих определять вероятности состояний системы в процессе ее функционирования и рассчитывать показатели эффективности по известным характеристикам входящего потока и каналов обслуживания.

Всеобщей аналитической модели для произвольной СМО не существует. Аналитические модели разработаны для ограниченного числа частных случаев СМО. Аналитические модели более или менее точно отображающие реальные системы, как правило, сложны и труднообозримы.

Аналитическое моделирование СМО существенно облегчается, если процессы, протекающие в СМО, марковские (потоки заявок простейшие, времена обслуживания распределены экспоненциально). В этом случае все процессы в СМО можно описать обыкновенными дифференциальными уравнениями, а в предельном случае, для стационарных состояний - линейными алгебраическими уравнениями и, решив их, определить выбранные показатели эффективности.

#### Пример Одноканальная СМО с ограниченной очередью

Если в очереди n мест для ожидания, то система может находиться в одном из следующих n+2 состояний:

S0- в системе нет заявок (ни в очереди, ни на обслуживании);

S1 - в системе обслуживается одна заявка, очередь пуста;

S2- в системе обслуживается одна заявка, и одна заявка находится в очереди, ожидает обслуживания;

…

Sn+1 - в системе обслуживается одна заявка и n заявок находятся в очереди, ожидают обслуживания.

Граф состояний такой системы представляет схему "гибели и размножения" ([рис. 2.12](http://www.google.com/url?q=http%3A%2F%2Fwww.intuit.ru%2Fdepartment%2Fcalculate%2Fcompmodel%2F2%2F4.html%23image.2.12&sa=D&sntz=1&usg=AFQjCNGa2Z8KSu8K5KMNmonYgS3cBz7oUQ)).

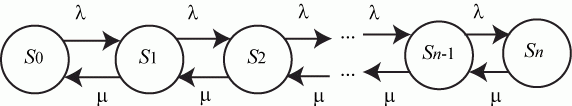


Рис. Граф состояний одноканальной СМО с ограниченной очередью

## 33. Пример имитационной модели СМО.

Имитационное моделирование представляет собой процесс построения и испытания некоторого моделирующего алгоритма, имитирующего поведение и взаимодействие исследуемой системы с учетом случайных входных воздействий и внешней среды.

Имитационная модель обладает самым главным свойством моделей вообще - она может быть объектом эксперимента, причем эксперимент проводится с моделью, представленной в виде компьютерной программы.

Имитационная модель отображает стохастический процесс смены дискретных состояний системы. При реализации модели на компьютере производится накопление статистических данных по показателям модели, которые являются предметом исследований. По окончании моделирования накопленная статистика обрабатывается, и результаты моделирования получаются в виде выборочных распределений исследуемых величин. Таким образом, математическая статистика и теория вероятностей являются математическими основами имитационного моделирования.

**Пример** Транспорт 1 с грузом отправился из пункта А в пункт С через пункт В. Одновременно из пункта D в пункт Е через пункт В отправился транспорт 2. Скорости движения транспортов распределены по нормальному закону с математическими ожиданиями V1 и V2 и стандартными отклонениями σ1 и σ2 .

Построить алгоритм имитационной модели (ИМ) с целью определения вероятности встречи транспортов 1 и 2 в пункте В. Расстояние от пункта А до пункта В S1 , а от пункта D до пункта В - S2.Событие встречи считать состоявшимся, если их времена прибытия в пункт В либо равны, либо отличаются на величину, не превышающую Δt.

Решение

Построим схему движения транспортов 1 и 2 ([рис. 3.2](http://www.google.com/url?q=http%3A%2F%2Fwww.intuit.ru%2Fdepartment%2Fcalculate%2Fcompmodel%2F3%2F%23image.3.2&sa=D&sntz=1&usg=AFQjCNEX8AcDXfAYbDhZxCvGEQ4ArsYsOQ)).

Возьмем две последовательности нормально распределенных случайных чисел:

V11, V12, …. , V1i, …..., V1N;

V21, V22, …., V2i, …., V2N;

характеристики которых соответствуют матожиданиям и стандартным отклонениям скоростей движения транспортов 1 и 2.

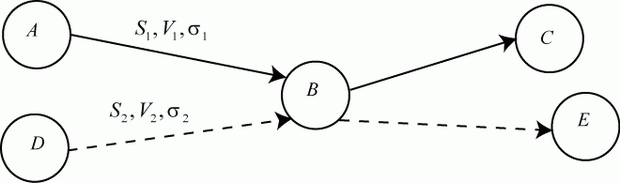


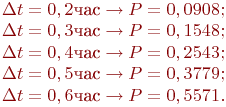
Рис. 3.2. Схема движения транспортов

1. Имитируем движение транспортов 1 и 2 до пункта В со скоростями V11 и V21 соответственно, взятыми из последовательностей нормально распределенных случайных чисел.
2. Вычислим время t1 и t2 прибытия в пункт В транспортов 1 и 2 соответственно:
3. 
4. Оценим результат имитации движения транспортов 1 и 2, т. е. установим факт наличия или отсутствия их встречи:
   * если , встреча состоялась;
   * если , встреча не состоялась.
5. Если встреча состоялась, зафиксируем этот факт увеличением значения на , т. е. M = M + 1(вначале M = 0).
6. Для нахождения вероятности встречи транспортов 1 и 2 повторим имитацию их движения N раз.

Рассчитаем вероятность встречи:



Результаты моделирования при N = 10000 и характеристиках движения транспортов: , , :



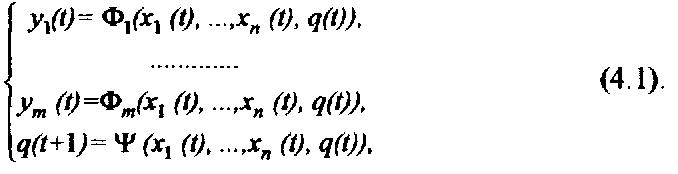
## 34. Общая характеристика уровня функционально-логического проектирования.

Модель ОП на функционально-логическом уровне проектирования обычно формулируется в терминах структурной теории автоматов. Логический элемент (автомат) может задаваться двумя способами.

**Первый способ** предполагает, что для описания объекта, имеющего внутренний алфавит *Q,* входные и выходные полюса xi, i = 1,n и yj, j = 1,m соответственно, необходимо:

• чтобы каждому входному полюсу xi, i = 1,n был приписан один и тот же входной алфавит *X,* аналогичным образом, каждому yj, j = 1,m - выходной алфавит *Y*;

• задать систему канонических уравнений



**Второй способ** предпологает что детерминированный элемент может быть задан как пятерка

<X, Y, Q, F, Y> , где

• *X, У, Q-* соответственно входной, выходной и внутренний алфавиты;

• F - функция выходов

• Y - функция переходов

Различают два вида схем: комбинационной принято называть схему, реализующую некоторый тривиальный оператор (набор булевых функций), а последователь­ностной - схему, реализующую некоторый автоматный оператор.

## 35. Задача синтеза на уровне функционально-логического проектирования.

Функционально-логический уровень проектирования, как и в любом другом этапе, содержит решение двух задач: синтеза и анализа

Задача синтеза решается при помощи теории цифровых автоматов.

Задача анализа решается при помощи методов логического моделирования.

Любая модель – функция от внутренних и внешних параметров.

**y = F(x,Q)**

Существуют аналитические, логические (имитационные) модели. Аналитические модели представляют собой описание функциональности устройства в виде системы уравнений. Решение данной системы позволяет получать характеристики проектируемого устройства.

При проектировании сложных систем аналитических моделей:

1. Сложность контроля исходных данных

2. Плохая приспособленность к формам графически-конструкторского описания схем.

3. Неприемлемость для анализа устройств, которые описаны в небулевском базисе.

4. Недостаточная компактность описания сложных объектов

Поэтому используют обычно метод логического моделирования. Процесс логического моделирования заключается в построении моделей объекта, описывающих его поведение и определение множества динамически изменяющихся состояний объекта и определение реакции на входные воздействия.

Основное требование к результатам анализа – обеспечение этих результатов истинному поведению исследуемого объекта.

Однако в связи с различными упрощениями, принятыми при построении моделей, можно говорить о степени соответствия модели объекту (адекватность модели и адекватность моделирования).

## 36. Задача анализа на этапе функционально-логического проектирования.

Цель задачи анализа: определить функциональность при заданном структурном и параметрических описаниях. Задача анализа на функционально-логическом уровне использует аппарат передаточных ф-ий для непрерывных моделей и аппарат матлогики и конечных автоматов.

Задача анализа схем сводится к двум задачам:

- статический анализ

- динамический анализ

В статическом анализе используются идеальные модели элементов схемы. Это позволяет проверить корректность процедуры синтеза.

**При решении задачи статического анализа решаются следующие задачи:**

-определение достижимости требуемого состояния

-установления закона функционирования схемы

-определения множества последовательности вх. Сигналов, вызывающую заданную последовательность внутри схемы и выходных сигналов.

-сравнение характеристик различных решений

-анализ вариантов системных решений для оптимизации

-проверка корректности межэлементных связей внутри схемы с учетом требований конкретных системных элементов.

**Динамический анализ определяет характеристики переходных процессов и решает задачи, дополнительные к статическому анализу:**

-определение параметров сигнала во время переходных процессов

-анализ частотных характеристик схемы

-определение алгоритмической устойчивости схемы

Если объект исследования – сложная система, то используется статистические модели (например СМО).

Проектирование:

1.составление ТЗ

2.Выполнить задачу синтеза

3.составить модель объекта

4.выполнить анализ

5. Результаты анализа удовлетворяют ТЗ (Ели «да» -составить ТЗ и переход на след. уров)

Если «нет» 6. Принять решение:

а)изменить параметры =>4

б)новый синтез =>2

в)перейти к предыдущ этапу.

## 37. Модели элементов в системе логического моделирования.

**E = (ϕ,А,Δ).**

А-алфавит в котором работает схема.

Δ-динамические параметры с задержками.

ϕ-функцион.

Логический элемент можно представить как 2 блока: логический и динамический. По степеням адекватности модели бывают :

1) **л-модели.**учитывают только логику , но не задержку =>позволяют определить достижимость определенного состояния.

2) **ЛД-модели**.(логика и динамика)

## 38. Модели сигналов в системе логического моделирования.

Различают статические состязания(предполагают появлене одного единичного (нулевого) сигнала на вых схемы, не предписанного законом функционирования т.е. если в момент времени т1 на вых появится у(т1) и в течении времени т1…т2 он не меняется, но в момент времени тау появл. Сигналы https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=y%28%5Ctau%7B%7D%29%5Cne%7B%7Dy%28%D1%821%29, то такие состязания статическими состязаниями т1< https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%5Ctau%7B%7D <т2.

**Динамическое состязание** – несанкционирование изменение сигналов на выходе несколько раз. Функцион. состязание возникает,если при переходе на входе от х1 к х2 возникает больше 2-ух алгоритмических переходов.

Логическое моделирование выполняется с пом. Системы логического моделирования – комплекс программных аппаратны, методических и информационных ф-лов, необходимых для лог. Моделиров.

ЗЫ не знаю сюда ли след. инфа, но на всяк случай она здесь.

Самая простая модель

– **двоичная**(чисто статическая модель) А2=(0,1).

для расширения возможностей такой модели добавляют третье состояние в котором мы не можем определить выходные состояния А3=(0,1,Х). А4=(0,1,Х,λ,Ε), λ-переключение сигнала 0->1,Ε переключение сигнала 1->0. А5=(0,1,Х,λ,Ε,Х). А6=(0,1,Х,λ,Ε,p,h). P-статический сбой, h-динамический сбой. А7=(0,1,Х,λ,Ε,p,h,Х) А2, А5…… А19. В последнем учитывается все изменения сигналов чем больше алфавит тем больше адекватность.

## 39. Модели схем в системе логического моделирования.

**Схемы** - это элементы и связи между ними.

Модели структур схем: в зависимости от **способа хранения информации** на схеме различают

1)компилятивные

2)интерпретативные

При этом различают 2 способа построения компилятивной модели:

1. описание модели выполняется на одном из специализированных языков(например VHDL

2. использование языков высокого уровня, при этом модель строится с учетом синтаксиса и семантики языка.

Первый способ удобнее, но второй – не требует знаний спец. языков

**Интерпретантивный** - табличное представление структуры схемы; при этом программа использует адреса связей перехода от одного эл-та к другому.Вычисление значение лог. Сигнала на выходе лог элемента сводится к извлечению из списков и их обработки. **Преимущество компететивной** – больше быстродействие. Недостаток – малая гибкость т.к. при вводе изменений в структуру схемы необх. повторная компиляция модели, тогда как для интерпретативной - просто вносим в списке указатели.

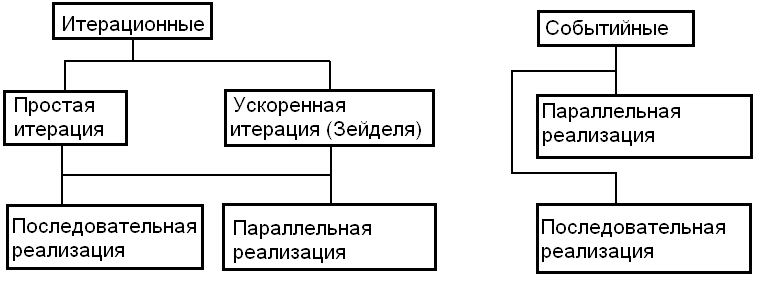
## 40. Риски сбоя в схемах и способы их обнаружения

## 41. Общая характеристика алгоритмов моделирования.

Существует много различных методов и алгоритмов моделирования.

Основными характеристиками алгоритмов моделирования являются адекватность, быстродействие и объем памяти, необходимый при реализации. Под адекватностью понимается степень соответствия ре­зультатов моделирования истинному поведению исследуемого ДУ.

Классификация:



**Преимущества и недостатки:**

**Последовательная реализация алгоритма простой итерации:**

преимущество: простота реализации;

недостаток: большое время вычисления, большие

вычислительные затраты.

**Параллельная реализация алгоритма простой итерации:**

преимущество: простота реализации, ускорение за счет меньшего числа обращений к описанию схемы;

недостаток: большое время вычисления, большие вычислительные затраты.

**Алгоритм ускоренной итерации (Зейделя):**

преимущество: ускорение за счет использования на текущей итерации значений, вычисленных на этой же итерации;

недостаток: необходимость предварительного ранжирования элементов логической схемы.

**Событийный алгоритм моделирования:**

преимущество: ускорение за счет просчета только тех элементов, у которых изменилось значение сигнала хотя бы на одном входе;

недостаток: необходимость выделения оперативной памяти для хранения таблицы текущих событий и таблицы будущих событий.

## 42. Алгоритм простой итерации (2-ая модель сигналов).

Существуют две реализации алгоритма простой итерации: последовательная реализация, параллельная реализация. При последовательной реализации алгоритма простой итерации необходимо задать начальное состояние схемы, затем подать первый входной набор и вычислить значения всех выходов логических элементов схемы на данном наборе. Если результат текущей итерации не равен результату предыдущей итерации, выполняется следующая итерация. Процесс заканчивается при полном совпадении результатов двух соседних итераций либо при достижении заданного максимального числа итераций. Максимальное число итераций в данном случае равно длине максимальной цепочки логических элементов (включая ОС) плюс 1. Также признаком конца процесса моделирования может служить наличие в таблице моделирования записи результата текущей итерации, находящейся на месте, отличающегося от предпоследнего. При параллельной реализации одновременно моделируют несколько входных наборов. Число одновременно моделируемых наборов зависит от языка программирования и особенностей реализации алгоритма. Ускорение происходит за счет меньшего числа обращений к описанию схемы.

Количество тактов (проходов) схемы (максимальное количество итераций) = самый длинный путь + 1. В данном алгоритме значение на любом элементе в схеме считается как функции, которая зависит от входного набора и значений элементов на предыдущей итерации:

https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=Y%7B%5C+%7D_%7Bij%7D%5E%7Bk%7D%5C+%3D%5C+F%5C+%28%7Bx%7D%5E%7Bk%7D%5C+%2C%5C+%7BY%7D_%7Bi-1%7D%29, где

x = {x1,x2, … xl}- входной набор (множество входных сигналов);

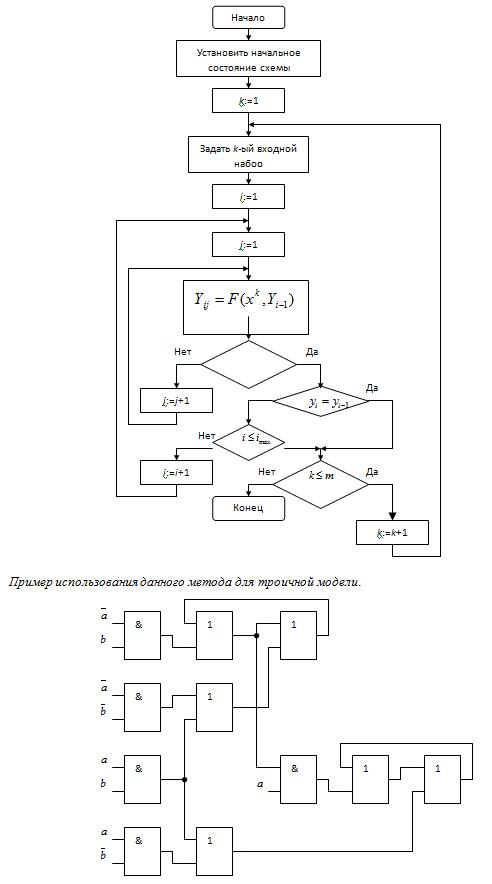
Y = {y1,y2, … yn}- выходные сигналы схемы;

*j*- номер элемента;

*i* - номер итерации;

*k* - номер входного набора (https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=k%5C+%3D%5C+%28%5Coverline%7B1..m%7D%29).

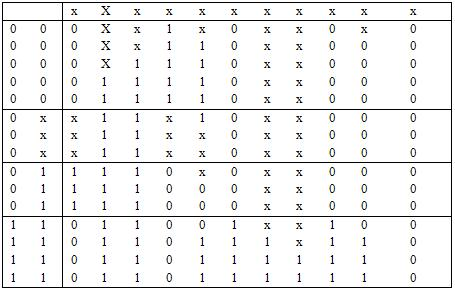
**Схема алгоритма**



Максимальное число итераций https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7Bi%7D_%7Bmax%7D= (1-2-3-2-6-7-8-7-8 – самый длинный путь).

Для последовательности наборов: 00, 0х, 01, 11.

Для троичной модели начальное значение на элементах не определено.



При использовании троичной модели в схеме можно обнаружить генераторы и состязания сигналов.Если при походе схемы максимальное число итераций значения на некоторых элементах в схеме не установились, то данные элементы образуют генератор.

Если на одном входном наборе в схеме значение на выходе элемента принимало определенное значение, после чего было не определено, а затем приняло тоже значение, то на данном элементе идет состязание сигналов.

## 43. Алгоритм ускоренной итерации (2-ая модель сигналов).

Необходимо определить начальное состояние схемы. При вычисле­нии значения на выходе i – того элемента на итерации h значения на вхо­дах выбираются следующим образом: если входная линия (линиями схемы будем называть внеш­ние входы схемы и выходы элементов) имеет номер j < i, используется ее значение, полученное на итерации h, а если j > i, — значение, полученное на итерации h - 1.При использовании итераций Зейделя требуемое число итера­ций зависит от порядка нуме­рации линий схемы. Поэтому используют алгоритмы ранжирования.

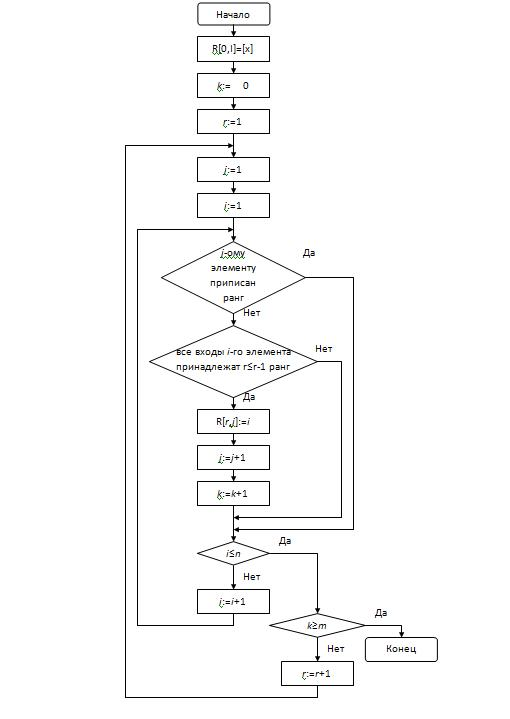
Количество тактов (проходов) схемы (максимальное количество итераций) = количество обратных связей + 2. При использовании данного алгоритма максимальное число итераций в последовательной схеме из 3 элементов, при изменении входного набора, будет равно 2 тактам, а не за 4 как в алгоритме простой итерации.

Для данного алгоритма:

https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=Y%7B%5C+%7D_%7Bij%7D%5E%7Bk%7D%5C+%3D%5C+F%5C+%28%7Bx%7D%5E%7Bk%7D%5C+%2C%5C+%7By%7D_%7Bi1%7D...%5C+%2C%5C+%7By%7D_%7Bij-1%5C+%7D%2C%5C+%7By%7D_%7Bi-1j%5C+%7D%2C...%5C+%2C%7B%5C+y%7D_%7Bi-1n%5C+%7D%29.

Если нумерация элементов в схеме неправильная нужно перенумеровать элементы.

**Схема алгоритма**



*r* – ранг;

*k* – количество элементов с рангом;

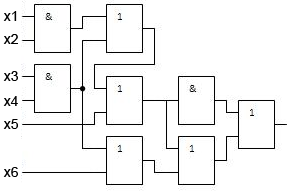
*i* – номер элемента;

*j* – номер элемента рангов;

*n* – количество элементов схемы;

R[*r*,*j*] – таблица рангов.

**Пример**



R0 – X1 X2 X3 X4 X5 X6

R1 – 1, 3 (*k*=2)

R2 – 2, 7 (*k*=4)

R3 – 4 (*k*=5)

R4 – 5, 8 (*k*=7)

R5 – 6 (*k*=8)

Данный алгоритм не работает для схемы с обратной связью. Для решения данной проблемы используется алгоритм условного ранжирования.

## 44. Алгоритм простой итерации (3-ая модель сигналов).

## 45. Алгоритм ускоренной итерации (3-ая модель сигналов).

## 46. Алгоритмы ранжирования.

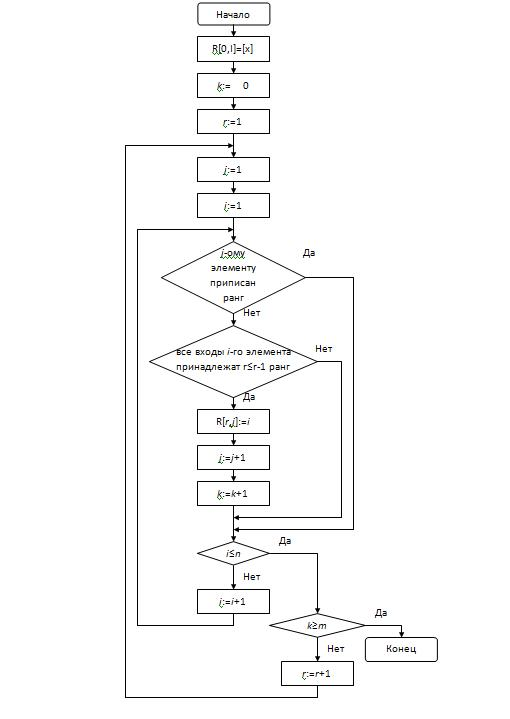
Метод ранжирования – Каждый из I экспертов расставляет n критериев по порядку (в порядке убывания значимости). На основе этой оценки каждому из параметров присваивается ранг, равный n-i. Это значение называется преобразованный ранг i-го критерия, тогда

**Сi = (k=1,L)(ri,k/(i=1,n)(k=1,L) ri,k).**

а) 0-й ранг – всем входным сигналам; 1-й ранг – всем элементам, связанным с элементами 0-го ранга; дальше – по алгоритму ручейка.

б) алгоритм условного ранжирования (для последовательностных схем не всегда точен): выполняется алгоритм ранжирования для всех элементов схемы, которым можно присвоить ранги; если есть ранг без элементов и элементы без рангов, то пустой ранг присваивается любому неотранжированному элементу схемы и продолжает выполнятся алгоритм ранжирования.

**Схема алгоритма**



*r* – ранг;

*k* – количество элементов с рангом;

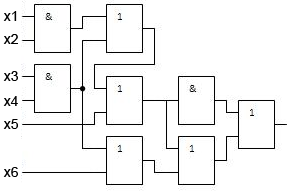
*i* – номер элемента;

*j* – номер элемента рангов;

*n* – количество элементов схемы;

R[*r*,*j*] – таблица рангов.

**Пример**



R0 – X1 X2 X3 X4 X5 X6

R1 – 1, 3 (*k*=2)

R2 – 2, 7 (*k*=4)

R3 – 4 (*k*=5)

R4 – 5, 8 (*k*=7)

R5 – 6 (*k*=8)

Данный алгоритм не работает для схемы с обратной связью. Для решения данной проблемы используется алгоритм условного ранжирования.

## 47. Событийный алгоритм моделирования (статическая модель элементов).

Значение сигнала на выходе элемента может измениться только в том случае, если изменилось значение сигнала хотя бы на одном его входе. Это обстоятельство лежит в основе алгоритмов событийною моделирования, где обработке подле­жат только элементы с изме­нившимися входными значения­ми сигналов.

При событийном алгоритме моделирования используются две таблицы:

**таблица текущих событий** (хранит номера элементов, которые необходимо просчитать в данный момент времени);

**таблица будущих событий** (хранит номера тех элементов, на входах которых произошли события).

Алгоритм событийного моделирования состоит в следующем:

- Задать в схеме начальное состояние.

- Задать установочный входной набор и просчитать схему на нем любым итерационным либо событийным алгоритмом, но в таблицу будущих событий записать все элементы схемы.

- Подать первый входной набор. В таблицу текущих событий записать номера входов, на которых произошли события, а в таблицу будущих событий записать те элементы, которые связаны с этими входами.

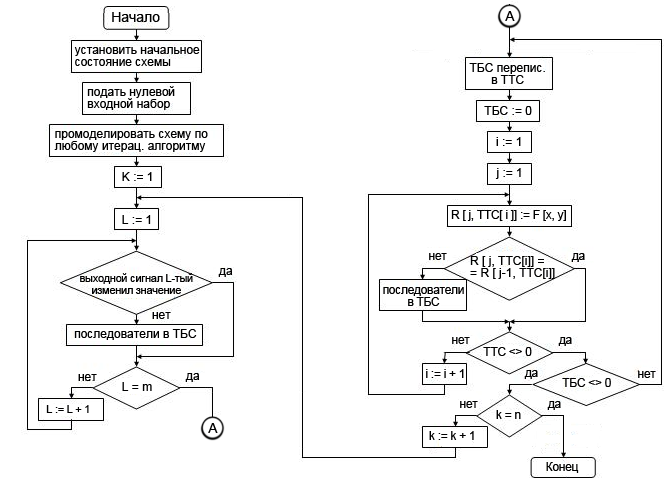
- Таблицу будущих событий переписать в таблицу текущих событий и очистить таблицу будущих событий. Просчитать все элементы из таблицы текущих событий.

- Последовательно моделируем элементы из таблицы текущих событий, при этом, если сигнал на выходе элемента изменился, то его последователя записываем в таблицу будущих событий. Данный процесс продолжается до тех пор, пока не станет пустой одна из таблиц. При генерации таблицы будут повторятся.

Событийный алгоритм:

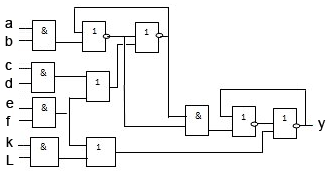
Выполняет только то, где возможны изменения, а не повторяет все, как итерационный алгоритм (то есть просчитываются только те элементы, на входах которых произошли изменения). Для этого нужна таблица будущих событий (ТБС) и таблица текущих событий (ТТС).

**Схема алгоритма**



На одном наборе меняют значение 30-40% элементов.

**Пример**

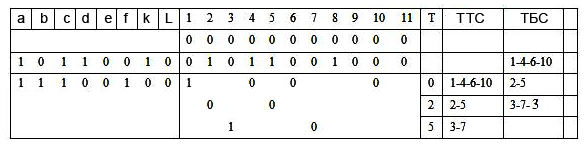


задержки

Тили = 1

Ти = 2

Тили-не = 3



## 48. Событийный алгоритм моделирования (ЛД – модель элемента).

Элементом называется конструктивно и функционально законченная часть устройства, не подлежащая дальнейшему расщеплению. В общем виде логические элементы описываются Е = {j, A, D}, где j - функция, А – алфавит, D - динамические параметры. Обязательно нужно задать j. Если D не задано, то модель – статическая. Если же элемент задан только E = {j}, то А = {0, 1} – это.

Любой элемент можно представить функциональным и динамическим блоком:

Такая модель предполагает временные характеристики. Если нет временных, то это аналитическая статическая модель (Л -модель).

Самое общее описание задержек – это задержка срабатывания. D - блок можно представить в виде задержки срабатывания, если у(t + tD) = ¦(x(t)). Задержка срабатывания tD предполагает, что модель элемента обладает совершенной задержкой, т.е. временем переключения из одного состояния в другое. Для увеличения адекватности блок D может быть расширен с учетом времени фронта и задержки распространения сигнала с входа на выход: tD = t фр+ tр. Т.к. передний и задний фронт отличаются по длительности, то tD = t фр01+ t фр10 + tр01 + tр10.



Это ЛД (логико - динамическая) модель.

Необходимо учитывать инерционные свойства:



Тогда tD = t фр01+ t фр10 + tр01 + tр10 + tи01 + tи10 . Это ЛИД – модель.

## 49. Параллельная реализация итерационных алгоритмов.

Смотрим в каждом алгоритме рассматривается параллельная реализация. Позже это будет выделено.

## 50. Последовательностная реализация событийных алгоритмов.

Смотрим 47 вопрос - он изначально ПОСЛЕДОВАТЕЛЬНОСТНЫЙ так что пусть ***идут в хуй*** с такими вопросами

## 51. Понятия «контроль» и «диагностика» в теории диагностирования СВТ. Методы контроля и диагностики.

Под контролем СВТ принято понимать процессы, обеспечивающие обнаружение ошибок в работе, вызванных отказом или сбоем.

Под диагностикой подразумевается процедура локализации неисправности объекта, т.е. установления того, какая часть объекта контроля (ОК) является неисправной. При диагностировании производится локализация неисправности на более низком иерархическом уровне, чем при контроле. В некоторых случаях под процедурой диагностики подразумевается анализ результатов измерений некоторой совокупности параметров, на основе которого могут приниматься решения о техническом состоянии объекта, области его функциональной применимости, режимах эксплуатации. Из многообразия направлений следует особо выделить *тестовый* *контроль* (ТК), функциональный *контроль* (ФК) и *параметрический контроль* (ПК).

Процедура тестового контроля осуществляется в специально отведенные для нее промежутки времени, в течение которых ОК не эксплуатируется по назначению. Достоинство ТК состоит в его универсальности, т.е. возможности с помощью аналогичных по сути методов реализовать контроль и диагностику на всех стадиях жизненного цикла изделия. Функциональный контроль, который часто называют *аппаратным* или *схемным*, не предусматривает генерации специальных тестовых воздействий. Этот вид контроля используется на стадии эксплуатации. Параметрический контрольпредполагает оценку состояния ОК по косвенным признакам, которые имеют интегральный характер и чаще всего выражаются непрерывными величинами

## 52. Избыточность и трудоемкость в процедурах контроля и диагностирования СВТ.

Дополнительные затраты, связанные с контрольно-диагностическими мероприятиями, можно характеризовать *избыточностью* и *трудоемкостью*. избыточности выделяют три вида: *аппаратурную*, *временную* и *информационную*. Под трудоемкостью следует понимать технико-экономический показатель, в общем случае зависимый от перечисленных видов избыточности и учитывающий совокупные затраты на разработку, производство и эксплуатацию средств контроля или диагностики.

**Временная избыточность** предполагает дополнительные затраты времени на выполнение контрольных операций. Применительно к тестовому и параметрическому контролю СВТ для выполнения этих операций отводится специальный отрезок времени, т.е. временная избыточность непосредственно не влияет на их производительность. Исключением являются частные случаи контроля, использующие повторное решение задачи с применением тех же самых или эквивалентных алгоритмов и сравнением полученных результатов, поскольку повторное решение может рассматриваться как процедура тестирования, кстати, позволяющая обнаруживать только сбои. При функциональном контроле временная избыточность может оказывать существенное влияние на производительность СВТ. Уменьшить затраты, связанные с временной избыточностью, можно за счет *параллельного* выполнения основных и контролирующих операций, т.е. аппаратурной избыточности, или использованием для контроля «остатков» времени при синхронной организации вычислений процесса.

**Аппаратурная избыточность** имеет место практически всегда (исключение, например, все тот же повторный счет) и определяется необходимостью применения дополнительной аппаратуры для реализации процедур контроля. В случае функционального контроля эта аппаратура «встроена» в объект контроля, в связи с чем такой вид контроля часто называют аппаратным или схемным.

**Информационная** избыточность свойственна всем без исключения методам контроля. Так, при тестовом контроле дополнительная информация необходима для хранения образов входных воздействий и эталонных реакций, а при параметрическом, как минимум, образа эталона. Применительно к функциональному контролю информационная избыточность используется в основе большинства методов и заключается в избыточном кодировании информации для проведения контроля и коррекции ошибок в процессе ее преобразований.

## 53. Общая характеристика функционального контроля СВТ.

Особое значение в процессе эксплуатации СВТ имеет высокая достоверность определения их технического состояния и своевременное обнаружение отказов и сбоев, поскольку «позднее» их обнаружение может слишком дорого стоить или приводить к неустранимым (ка­та­стро­фическим) последствиям. Эта задача решается с помощью средств автоматического (встроенного) функционального контроля, которые при появлении ошибки немедленно приостанавливают вычислительный процесс и инициируют работу средств восстановления после сбоя, например повторное выполнение операции. В случае обнаружения отказа средства функционального контроля могут инициировать работу автоматической системы тестового диагностирования, локализующей неисправность.

Если рассматривать процесс вычислений как информационный, то все составляющие его операции могут быть разделены на три типа: операции передачи, логические и арифметические преобразования. Следовательно, для функционального контроля необходимо иметь средства проверки правильности выполнения всех типов операций и, по возможности, средства коррекции ошибок

Вопросы, связанные с безошибочной передачей информации настолько важны, что их исследование являются предметом самостоятельной теории – теории кодов, контролирующих ошибки [12]. Техническая задача, решаемая с помощью этой теории в общем случае состоит в защите цифровых данных от появляющихся в процессе передачи по каналам связи ошибок.

Применительно к функциональному контролю СВТ, независимо от типа операций, в большинстве случаев удобно использовать модель, основанную на применении равномерных избыточных кодов, которые в дальнейшем называются *разделимыми*. Особенностью этих кодов является то, что кодовое слово можно разделить на два подслова фиксированной длины. Первое подслово является самой преобразуемой информацией (в дальнейшем это подслово будет называться кодом информации). Второе – ее кодирующим отображением (контрольным кодом) в соответствующем алфавите, причем это отображение, в общем случае, не взаимно однозначно.

С учетом кодирования информации разделимым избыточным кодом при функциональном контроле обычно используется универсальный прием, который сводится к выполнению следующих инструкций:

- получить в соответствии с некоторым правилом кодирующее отображение для подслова, являющегося контрольным кодом преобразуемой информации, т.е. сформировать разделимый код;

- раздельно выполнить эквивалентные операции преобразования над кодом информации и контрольным кодом;

- выполнить повторное формирование контрольного кода применительно к преобразованному коду информации (получить контрольный код результата);

- сравнить результат выполнения предыдущего шага с результатом преобразования контрольного кода;

- если результат сравнения положителен (отличий в контрольных кодах нет), то полагать, что ошибка отсутствует, иначе инициализировать аварийные действия.

При формализации модели построения контрольных кодов применительно к обработке информации средствами вычислительной техники удобно использовать отображение кода информации в контрольный код, соответствующее представлению чисел в системе счисления в остатках (системе остаточных классов – СОК). Представление чисел в этой системе счисления определяется следующим.

Пусть некоторое целое неотрицательное число А, заданное в позиционной однородной системе счисления, представимо в виде:

https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=A%3Daq%2B%7Br%7D_%7Ba%7D, где где a,q,https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7Br%7D_%7Ba%7D– целые неотрицательные числа и https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7Br%7D_%7Ba%7D< q

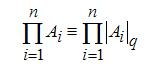
Теорема 2.1.

Сумма чисел https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7BA%7D_%7Bi%7D%2C%5C+i%5C+%3D%5C+%5Coverline%7B1..n%7D по модулю q с суммой остатков https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7B%5Cleft%7C%7B%7BA%7D_%7Bi%7D%7D%5Cright%7C%7D_%7Bq%7D%5C+i%5C+%3D%5C+%5Coverline%7B1..n%7D%5C+%D1%8D%D1%82%D0%B8%D1%85%5C+%D0%B6%D0%B5%5C+%D1%87%D0%B8%D1%81%D0%B5%D0%BB:



Теорема 2.2.

Произведение чиселhttps://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7BA%7D_%7Bi%7D%2C%5C+i%5C+%3D%5C+%5Coverline%7B1..n%7D сравнимо по модулю q с произведением остатковhttps://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7B%5Cleft%7C%7B%7BA%7D_%7Bi%7D%7D%5Cright%7C%7D_%7Bq%7D%5C+i%5C+%3D%5C+%5Coverline%7B1..n%7D%5C+%D1%8D%D1%82%D0%B8%D1%85%5C+%D0%B6%D0%B5%5C+%D1%87%D0%B8%D1%81%D0%B5%D0%BB



**Теорема 2.3.**

Для заданного множества целых положительных попарно взаимно простых чисел m1, m2, ..., mk, и множества неотрицательных целых чисел c1, c2, ..., ck, при ci<mi, система сравнений



имеет не более одного решения с в интервале ,



## 54. Общая характеристика параметрического контроля СВТ.

Системы параметрического контроля строятся на основе оценки состояния ОК по некоторым косвенным признакам и обладают неоспоримым преимуществом, выражающимся в не повреждающем характере испытательных воздействий, которые в явном виде, как правило, не используются. При тестовом контроле для доступа к внутренним точкам ОК часто приходится применять специальные игольчатые контактные устройства, которые часто называют ложем из гвоздей (bed of nails), и специальные меры для "отключения" отдельных компонент от остальной части ОК, именуемые применительно к электронным системам «электронными ножами» (electronic knife).

Второе преимущество параметрического контроля заключается в следующем. Обычно тесты предназначаются для выявления одиночных неисправностей (под одиночной понимается неисправность с точностью до одного элемента, на выходе которого она может быть зафиксирована). Задача построения тестов для выявления и локализации кратных неисправностей, т. е. одновременно присутствующих в объекте двух или более неисправностей, а также неисправностей, возникающих из-за постепенного ухудшения, например, формы сигналов в настоящее время за исключением простейших ситуаций относится к классу сложных задач диагностики. Контроль и диагностика объектов с такими неисправностями должен производиться по специально разработанным методикам, позволяющим по признакам одиночных неисправностей локализовать группу неправильно функционирующих элементов или цепь с затухающим сигналом.

Параметрической контроль в равной мере и без дополнительных затрат позволяет обнаруживать как одиночные, так и групповые неисправности, однако, оказывается ориентированным на класс неисправностей, которые могут устанавливаться только по косвенным признакам, что может быть отнесено к его недостаткам.

Например, для электронной аппаратуры отличие температурных или физических полей ОК от эталонных для заданного конструктивного узла не позволяют определить характер самой неисправности и классифицировать ее в рамках, традиционных для этого класса объектов (одиночные константные и инверсные неисправности на входных и выходных полюсах элементов, «ближайшее соседство» – паразитная связь между элементами и т.п.). Кроме того, параметрический контроль в общем случае не позволяет определить неисправности, связанные с динамическими характеристиками объекта («паразитная задержка» –увеличение времени прохождения сигналов через устройство) или его «чувствительности к наборам (инструкциям)» – неправильном функционировании, вызванном определенным сочетанием значений входных переменных (неправильное функционирование программируемых устройств при выполнении определенных команд) и др.

Отмеченный недостаток частично компенсируется тем, что параметрический контроль (диагностика) не требует детального структурного описания ОК. Действительно, при решении задач контроля приходится учитывать следующий факт: чем сложнее объект, который необходимо диагностировать, тем меньше информации о его структуре доступно специалисту по диагностированию. Поэтому при создании тестовых программ для таких объектов разработчик вынужден предполагать, что неисправность может привести к реализации любой функции, отличной от заданной. Формализация такого класса неисправностей для тестового контроля достаточно сложна, а для параметрического контроля обычно не вызывает особых затруднений.

Кроме того, параметрический контроль не требует решения задач тестопригодного проектирования ОК, т. е. не требует при проектировании учета таких характеристик объекта, как «управляемость» и «наблюдаемость»

Достаточно широкий класс систем параметрического контроля строится на основе так называемых изображающих систем, обобщенная схема которой приведена на pис.

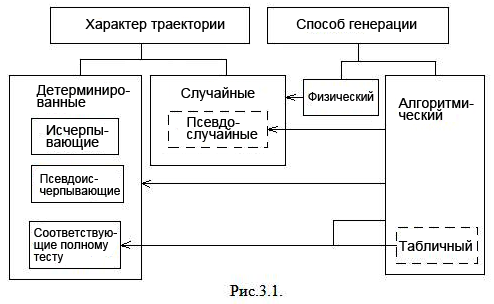
Система позволяет получить визуальную информацию, например о температурных или электрических полях, излучаемых ОК, и на основе анализа полученной информации на соответствие ее эталонной принимать решение о состоянии объекта или локализовать область неисправности.

## 55. Общая характеристика тестового контроля СВТ.

Организация тестового диагностирования связана с наличием специально подготовленных воздействий и эталонных реакций на эти воздействия, позволяющих определить техническое состояние ОД и, в случае его неисправности, локализовать место неисправности, а также специальной аппаратуры, предназначенной для генерации входных воздействий и регистрации выходных реакций ОД с их последующим анализом

На рис.3.1 представлена классификация испытательных последовательностей. По характеру траектории они могут быть разделены на детерминированные и случайные. Детерминированные последовательности характеризуются тем, что к моменту начала тестового эксперимента их траектория однозначно определена. К ним можно отнести исчерпывающие, псевдоисчерпывающие и соответствующие понятию полного теста последовательности.

Исчерпывающая последовательность представляет собой полный перебор всех возможныхвходных воздействий. Следовательно, с ее помощью реализуется тест https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%D0%A2%5C+%5Csubseteq%7B%7D%5C+%7BC%7D%5E%7BY%7Dx%5C+%7BC%7D%5E%7BY%7D , который априорно обладает полнотой.



В большинстве случаев ОК можно разделить на ряд независимых частей, каждую из которых можно рассматривать как отдельный объект контроля. Если число входов для каждого из них оказывается меньше, чем для всего устройства в целом, то длина исчерпывающей последовательности может быть сокращена. Такая последовательность называется псевдоисчерпывающей и , по сути, является исчерпывающей для каждой из частей ОК, тестирование которых может производиться независимо друг от друга.

Ещё умные слова – функции счета, контрольные суммы, Синдромное тестирование, спектральн. коэфф.

## 56. Контроль передачи информации.

Как уже отмечалось, для контроля процесса преобразования информации средствами вычислительной техники, в основном, используются равномерные избыточные коды. Очевидно, что избыточность равномерного разделимого кода определяется длиной подслова, соответствующего контрольному коду. Применительно к СВТ широкого назначения эту избыточность стремяться по возможности минимизировать, даже в ущерб обнаруживающей способности, в силу следующих причин.

При формировании избыточного кода можно использовать два приема (иногда оба приема совмещаются). Первый из них предполагает формирование контрольного кода непосредственно при вводе исходной информации или при получении промежуточных (конечных) результатов вычислений и хранение в памяти всего избыточного кода. Объемы обрабатываемой информации для СВТ сегодня измеряются десятками-сотнями мегабайт и, следовательно, при длине контрольного кода, равном единице, потребуется аналогичное количество физических элементов памяти, высокие надежностные характеристики которых так же нужно обеспечивать. С увеличением длины подслова контрольного кода аппаратурные затраты будут кратно возрастать. Однако, несомненным достоинством рассматриваемого приема является возможность контроля операций записи-чтения – частного случая передачи информации из регистра числа в ячейку памяти и наоборот (по отношению к СВТ вместо «передача» чаще используют термин «пересылка»).

Во втором случае информация храниться в памяти в неизбыточном коде. Последний формируется после операции чтения с помощью соответствующего кодера и используется для контроля дальнейших пересылок. При записи информации в память контрольный код игнорируется. Требования к минимизации аппаратурной избыточности в этом случае менее жесткие, но возможность контроля операций записи-чтения теряется.

Не менее существенной причиной применения кодов, обнаруживающих только ограниченное множество ошибок, является то, что физическая природа ошибок для аппаратуры СВТ достаточно изучена и вероятность возникновения кратной ошибки при выполнении операций обработки информации, в отличие от передачи по каналу в системах связи, принимается небольшой.

*Код с проверкой четности*, который широко используется для контроля операции установки равенства или пересылок, образуется путем дополнения информационного подслова контрольным с длиной, равной единице, т.е. одного контрольного разряда. В соответствии с рассматриваемой моделью, информационное подслово здесь рассматривается как код числа в двоичной позиционной однородной системе счисления (в дальнейшем просто двоичной), а контрольный код его отображением в СОК. Такой способ кодирования обычно называют *числовым* и контроль с его применением – *числовым* *контролем* по модулю. По определению оба подслова есть код в двоичном стандартном алфавите. Следовательно, единственно возможным модулем для представления контрольного кода является число «2». Значение модуля совпадает с основанием двоичной системы счисления, в котором представлено информационное подслово, и обнаруживающая способность контрольного кода оказывается ничтожной, поскольку с его помощью можно установить только ошибку в младшем разряде двоичного числа. Иными словами один контрольный разряд позволяет определить только четное и нечетное значения кода информации, а остальные ошибки обнаружены не будут.

Существенно увеличить обнаруживающую способность кода с проверкой четности можно, если изменить вид алфавитного отображения, используя для представления в СОК по модулю «2» *сумму* содержащихся в двоичном представлении информации единиц или нулей. Тогда код позволит обнаруживать все одиночные и кратные нечетным ошибки. Поскольку единицы или нули могут рассматриваться как цифры числового представления, такой контроль часто называют «цифровым контролем по четности».

Код Хемминга строится таким образом, что к имеющимся информационным разрядам слова добавляется определенное число контрольных, которые формируются перед передачей путем подсчета четности для определенных групп информационных разрядов.

Требуемое число контрольных разрядов определяется следующим образом.

Пусть кодовое слово длиной n разрядов имеет m информационных и k = n - m контрольных разрядов. С помощью k разрядов контрольного слова можно представить https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7B2%7D%5E%7Bk%7Dподслов, соответствующих отсутствию или наличию ошибки в информационных разрядах. Таким образом ,

https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7B2%7D%5E%7Bk%7D%5C+https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%3E%3D%5C+n%2B1%5C+или https://www.google.com/chart?cht=tx&chf=bg,s,FFFFFF00&chco=000000&chl=%7B2%7D%5E%7Bk%7D- k - 1>= m .

Контроль с помощью кода Хемминга реализуется с помощью набора схем подсчета четности, которые при кодировании определяют контрольные разряды, а при декодировании – корректирующее слово.

Наибольшее применение этот код нашел при контроле в запоминающих устройствах

## 57. Метод таблиц истинности синтеза тестов.

Для выявления неисправности надо промоделировать схему и сравнить ее с эталоном. B = {B0, Bx}. B0 – модель исправного объекта, Bx = {b1, …,bn}-модели неисправностей. Необходимо промоделировать все b модели и выделить те проверки, которые обнаруживают данную неисправность. Для получения минимального теста используют импликантную матрицу. При тесте комбинационной схемы последовательность подания входных наборов не имеет значения. При тесте последовательностных схем, последовательность должна выстраиваться по последовательности изменений внутренних состояний схемы. Перед поданием тестовой последовательности необходимо установить схему в нужное состояние с помощью установочной последовательности. Установочная посл.+ тестовая = сегмент теста.

## 58. Метод активизации путей синтеза тестов.

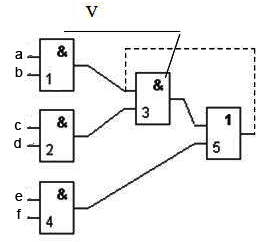
Метод активизации путей – необходимо задать неисправнyt’

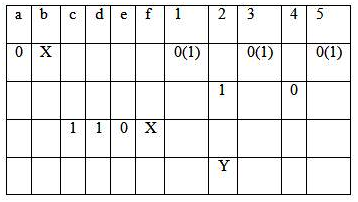
ость.

**1-я часть** – активизация пути

**2-я часть** – дополнение входного набора

Активизируемый путь – это путь, по которому данная неисправность может быть передана на выход схемы.



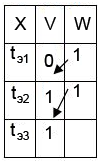


tэ =0Х110Х 0(1)

Если кратные неисправности, то не все можно обнаружить. Если схема с обратными связями, то ее преобразовывают по модели Хаффмана – разрывают ОС (пунктиром) и добавляют вход (V).

tэ =0Х110Х y=1

Установочная последовательность – подается до того, как подается тестовая последовательность. Она устанавливает схему в нужное состояние, но сама тестом не является.



V- вход, W – выход. Нужно найти такой тест, у которого на входе 1 нужна (W=1, с предыдущего теста).

Сегмент теста – последовательность упорядоченных по изменению внутренних состояний схемы.

Установочных последовательностей нужно столько, сколько будет сегментов.

## 59. Д-алгоритм синтеза тестов.

## 60. Метод частной булевой производной синтеза тестов

Булевой производной функции

f(x) = f(x1, x2,..., xn) по xi называется функция

df(x) / dxi = f(x1, x2,..., xi,..., xn) Å f(x1, x2,...,nxi,..., xn), где Å – сумма по модулю 2.

Булева производная может быть также вычислена и по следующей формуле:

df(x) / dxi = f(x1, x2,..., 0,..., xn) Å f(x1, x2,..., 1,..., xn).

Булева производная определяет значения логических переменных x1,..., xn (кроме xi), при которых изменение состояния xi приводит к изменению значения функции f(x).

Тест для неисправности xi = 0 (хi = 1) определяют значения логических переменных, при которых

xi × df(x) / dxi = 1 (`xi × df(x) / dxi =1).

Сказанное можно распространить и на внутренние переменные. Тест для неисправностей z=0 (z=1) внутренней линии схемы определяют значения логических переменных, при которых

z× df(x) / dz =1 (nz× df(x) / dz = 1).

Таким образом, входное воздействие для проверки неисправности в точке z определяется следующим образом.

1. Составляем функцию f(x), в которой в качестве переменной присутствует z.

2. Определяем частную булеву производную df(x) / dz, приводим полученное выражение к дизъюнктивной форме (ДФ).

3. Выбираем один из термов (например, t), полученной в п. 2 ДФ.

4. Неисправность z = 0 проверяется на воздействии, при котором значения переменных x1,..., xn обеспечивают условие z× t = 1.

5. Неисправность z = 1 проверяется на воздействии, при котором значения переменных x1,..., xn обеспечивают условие `z× t = 1.

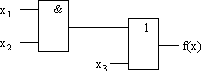
Приведем несколько примеров вычисления тестов методом булевой производной.

Пример. Дана схема (рис. 4.6), реализующая функцию

f(x)= x1 х2 Ú x3. Найти тесты неисправностей x1 = 0 и x1 = 1.

Найдем булеву производную df(x) / dx1:

.



Тест для x1 = 0 определим из условия x1× df(x) / dx1 = 1, то есть . Следовательно, x1 = 1, x2 = 1, x3 = 0.

Тест для x1 = 1 определим из условия , то есть 

Следовательно, x1 = 0, x2 = 1, x3 = 0.

## 61. Цепной метод поиска булевой производной.

## 62. Применение префиксной формы задания функции в цепном методе поиска булевой производной.

## 63. Метод эквивалентных нормальных форм синтеза тестов.

## 64. Синтез тестов для последовательностных схем.

Все алгоритмы предназначены для построения тестов для комбинационных схем, поэтому используют модель Хадтмана: разрыв обратной связи и добавление дополнительных входов и выходов ОС.

Вх. воздействия в цепях ОС: V = {V1,…,Vk}

Вых. воздействия в цепях ОС: W = {W1,…,Wk}

V(t + t) = W(t); Y(t) = F(x(t), V(t - t)); W(t) = j(x(t), V(t)).

Последовательность проверок определяется последовательностью внутренних состояний.

Сущ. 2 способа таких тестов:

1) синтез тестов при помощи моделирования. С помощью исправной схемы получаем таблицу истинности на всех наборах и внутр. состояниях. Далее вносится неисправность и моделирование повторяется. Полученные две таблицы сравниваются и наборы, у которых изменилось значение, будут тестовыми, но нужно найти установочную последовательность, обеспеч. работу этой комбинации.

2) основан на алгебре Рота.

## 65.Контроль арифметических и логических операций.

Для контроля арифметических операций обычно используется числовой контроль по модулю q. Возможность проверки правильного результата выполнения операций сложения и умножения с помощью контрольного кода, представленного в СОК, вытекает из теорем 2.1 и 2.2.

При контроле операции деления используется зависимость вида a=c\*b+r

где а – делимое, с – частное, b – делитель и r – остаток от деления, т.е. в процедуре контроля сравнивается контрольный код делимого с результатом вычисления правой части выражения, содержащего только допустимые в СОК операции.

На рис.2.2. представлена структурная схема операционного блока с числовым контролем по модулю (схему можно модифицировать с учетом контроля операции деления, что предлагается сделать в качестве упражнения). Ее аппаратурная избыточность определяется схемами формирования контрольного кода и контролирующего операционного блока, сложность которых зависит от значения q .

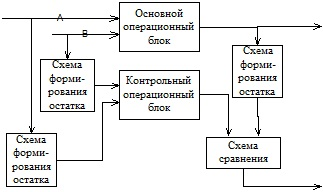


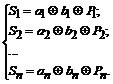
рис 2.2

Этим же значением определяется полнота контроля. При увеличении значения модуля увеличивается число кратных ошибок, обнаруживаемых системой контроля, однако, при этом возрастает и сложность кодирующей и контролирующей аппаратуры. Например, в качестве модуля q не следует выбирать основание системы счисления, используемой для представления основной информации, если она является однородной и позиционной. Это приводит к тому, что контролируются только младшие разряды чисел (последнее уже отмечалось). Избыточность будет минимальной при условии q=r+-1, где r– основание системы счисления, т.е. для СВТ значение q принимается равным «3». Ограничения на класс обнаруживаемых ошибок для СВТ широкого назначения при этом считаются приемлемыми .

Не смотря на то, что аналоги теорем для суммы цифр операндов и результата сложения и умножения отсутствуют, для контроля арифметических операций можно использовать цифровой контроль, что поясняется следующим примером.

Пусть контрольным кодом является остаток по модулю 2 суммы единиц в каждом из операндов.

При сложении чисел a и b разряды суммы s образуются следующим образом:



где – соответственно разряды суммы, операндов и переноса; – операция сложения по модулю 2.

Тогда, сложив все n равенств по модулю 2  .т.е. при правильном результате сложения четность суммы его цифр должна совпадать со значением, вычисленным согласно правой части приведенного выражения.

Применение такого контроля может быть оправдано специализацией СВТ, например, в случае, когда в системе команд предусмотрена только операция сложения и для контроля пересылок используется цифровой контроль по модулю 2. Контроль логических операций

К логическим принято относить операции сдвига, а также операции and, or, not, xor. Последние, начиная с and, могут применяться либо к так называемым логическим значениям true (истина) и false (ложь), кодами которых при n-разрядном представлении являются единица в младшем разряде при остальных нулях и все нули соответственно, либо ко всем разрядам произвольных операндов поразрядно.

Модели, с помощью которых могли бы строиться избыточные коды для контроля логических операций, вообще говоря, отсутствуют. Для теории кодов, исправляющих ошибки, такая задача (впрочем, как и контроль арифметических операций) не является предметом анализа. Применение используемой выше модели, точнее построение контрольного кода как отображения в СОК, жестко ограничено теоремами 2.1 и 2.2 (здесь «жестко ограничено» означает, что, например, для контроля сдвига такая возможность есть, поскольку операция сдвига – это умножение на где p – основание системы счисления, а n – количество разрядов, на которое сдвигается код информации). Перечень подобных «экзотических» приемов можно продолжить, рассматривая операцию *xor* как поразрядную сумму операндов по модулю «2» без учета переносов и корректируя в соответствии с этим выражение (2.1), что, однако, не применимо к другим операциям, и т.д.

Единственным очевидным обобщением может быть рекомендация строить избыточный код как разделимый. В этом случае всегда остается возможность использовать в качестве контрольного кода копию кода информации, т.е. дублировать информацию (собственно, представление контрольного кода в виде отображения ее в СОК – это то же дублирование, но не полное и со «сжатием» копии).

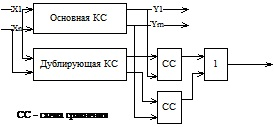


рис 2.3

## 66.Требования, предъявляемые к тестам.

*Достаточная полнота контроля*. Полный тест – тест, реализация которого позволяет обнаружить все возможные неисправности из заданного класса неисправностей Количественная оценка полноты контроля есть отношение количества выявленных неисправностей к общему числу возможных неисправностей заданного класса.

*Приемлемое время испытаний объекта контроля.* Время контроля непосредственно связано с количеством элементарных проверок или *емкостной* сложностью теста прямой (не обязательно линейной) пропорцией. Поэтому наиболее выгодно применение *минимальных* тестов – полных тестов, имеющих минимально возможное количество элементарных проверок, или, по крайней мере, не избыточных тестов – полных тестов, исключение из которых любой элементарной проверки нарушает полноту теста.

*Малая трудоемкость генерации тестов*. Создание тестов, эффективных по показателю полноты и количеству элементарных проверок, представляет собой сложную задачу. Ее решение для большинства аппаратных компонент СВТ возможно только с помощью автоматизированных систем генерации тестов (АСГТ).

*Глубина диагностирования*. Очевидным требованием к диагностическим тестам является локализация места неисправности с точностью до конкретного сменного элемента или до определения области применимости с ограничением функций для «неразборных» объектов, например БИС. При этом необходимо отметить, что в процессе проектирования систем технического диагностирования (СТД) необходимо, чтобы:

n алгоритмы технического диагностирования обеспечивали в первую очередь требуемую достоверность результатов определения технического состояния объектов диагностирования при заданных ограничениях на другие параметры СТД;

n проектирование СТД осуществлялось одновременно с проектированием объекта диагностирования;

n при проектировании объектов диагностирования предусматривалось обеспечение контролепригодности и достаточной глубины их диагностирования;

n способ генерации входных последовательностей (см ниже) и метод диагностирования соответствовали аппаратуре контроля.

## 67.Классификация методов тестового контроля.

Разделение на две составляющие – *входную* и *выходную* последовательности.

Входная последовательность определена на  – множество входных воздействий ( n– количество элементарных проверок), а элементы этого множества  есть векторы значений входных воздействий, упорядоченные по условным номерам входов объекта. Порядок следования входных воздействий задает траекторию изменения их значений на каждом из входов или группы входов контролируемого объекта во *времени*. Время здесь подразумевается автоматным, т.е. 

Входную последовательность удобно интерпретировать матрицей ,*,* каждая строка которой соответствует элементарной проверке, а, например, *k*-ый столбец отражает порядок следования (траекторию изменения значений) входных воздействий на *k*-том входе ОК. Такая матрица, по сути, представляет собой табличный способ задания входной последовательности и не всегда определяется в явном виде, поскольку значения ее элементов может быть вычислено по некоторому закону.

Аналогичным образом выходную последовательность можно представить в виде матрицы , ,где - количество выходов контролируемого объекта. Строка матрицы отражает состояние объекта в момент времени t , а столбцы – траекторию изменения значений реакции на каждом из выходов ОК в пространстве времени проверки. Регистрация реакций в этом случае может оцениваться как отображение ,или как некоторая интегральная величина, соответствующая каждому выходу по всему множеству например, в случае применения функций счета, сигнатурных анализаторов или синдромных сверток). При этом очевидно, что упорядоченность выходной последовательности можно не принимать во внимание, так как она однозначно определяется порядком следования входных воздействий.

В соответствии с такой моделью методы тестового диагностирования можно классифицировать в зависимости от *сочетания* способов получения испытательных последовательностей и способов оценки и регистрации реакций контролируемого объекта.

## 68.Методы сжатия реакций ОК.

Принятие решения о техническом состоянии ОК основывается на анализе выходной последовательности в матричном представлении .Очевидно, что для хранения матрицы необходимы большие объемы памяти, что объясняет важность методов сжатия выходной последовательности (диагностической информации). Методы, которые используют такое сжатие называют методами компактного тестирования.

Поскольку методы сжатия диагностической информации слабо зависят от вида испытательной последовательности, то они могут рассматриваться отдельно.

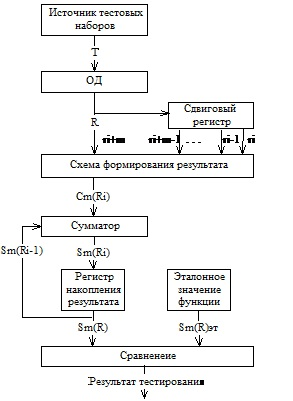
В общем случае, задача сжатия выходной информации связана с выбором кодирующего отображения, которое ставит в соответствие последовательности  код .Если в качестве последовательности w выбраны строки матрицы Y, то говорят о пространственном, а в случае, когда w соответствуют столбцы Y– о временном сжатии диагностической информации.

Поскольку эффект сжатия наблюдается только в случае то, естественно, возникает вопрос о достоверности компактного тестирования, т.е. о вероятности того, что примененный метод сжатия позволяет обнаружить все предполагаемые неисправности ОК. При этом анализируется влияние, которое оказывает неисправность на вид выходной последовательности и то, каким образом это влияние отображается в код K.

## 69. Функции счета.

На одновыходное тестируемое дискретное устройство подается последовательность тестовых наборов T.Сдвиговый *m*-разрядный регистр хранит *m* последних результатов

*Ri={ri, ri+1,..., ri+m-1,}* Величина *m³0* задает число последовательных результатов, подвергаемых анализу с целью определения наличия и числа представляющих интерес признаков сигналов в подпоследовательности *Ri* последовательности результатов *R.* Анализ осуществляет схема формирования результата, на выходы которой параллельно поступает последовательность результатов *ri, ri+1,..., ri+m-1,* а на выходах формируется текущее значение *Cm(Ri)* соответствующей функции счета. Сумматор вычисляет текущее суммар­ное значение *Sm(Ri)* функции счета путем арифметического сложения *Cm(Ri)* с хранящимся в регистре накопления результатов предыдущим суммарным значением *Sm(Ri-1)* функции счета. Окончательное значение функции счета сравнивается с эталонным значением *Sm(R)ЭТ* этой функции.



**рис 3.2**

Функция счета характеризуется глубиной памяти *т* (числом разрядов *A*) и видом признаков результатов. Наиболее просто реализуется тестирование на основе функций счета, имеющих глубину памяти 0 или 1. Такими функциями являются:

для *т=0*:

а) функция счета единичных значений результатов 

б) функция счета числа переходов изменений значений результатов из 0 в 1 и из 1 в 0



для *т=1*:

в) функция счета числа повторений значений результатов



г) функция счета числа фронтов (изменений из 0 в 1)



д) функция счета числа срезов (изменений из 1 в 0)



Для компактного тестирования многовыходных ДУ каждому получаемому на выходе ДУ двоичному набору *vi* ставится в соответствие его двоичный вес *аi.* В качестве функций счета применяются функции







где *V –* последовательность выходных наборов *v0, v1,...,vm; P –* предикат сравнения весов соседних наборов, принимающий значение 1, если результат сравнения совпадает с заданным предикатом, и значение 0 в противном случае.

## 70. Контрольные суммы.

Контрольные суммы- это один из основных методов сжатия выходной информации.

В общем случае, задача сжатия выходной информации связана с выбором кодирующего отображения, которое ставит в соответствие последовательности  код  (Если в качестве последовательности w выбраны строки матрицы Y, то говорят о пространственном, а в случае, когда w соответствуют столбцы Y– о временном сжатии диагностической информации.

Поскольку эффект сжатия наблюдается только в случае когда w соответствуют столбцыY– о временном сжатии диагностической информации.

Поскольку эффект сжатия наблюдается только в случае m<n, то, естественно, возникает вопрос о достоверности компактного тестирования, т.е. о вероятности того, что примененный метод сжатия позволяет обнаружить все предполагаемые неисправности ОК

**Контрольные суммы.** При использовании контрольных сумм совокупность результатов тестирования рассматривается как массив чисел, над которым выполняется операция поразрядного или арифметического суммирования

Пусть задано упорядоченное множество из *п* *m*-разрядных чисел {*ui*}, где *i=l,2,...,n*; *ui=ui1,ui2,...,uim* соответствующее выходной последовательности ДУ. Используются следующие способы суммирования:

а) поразрядное суммирование по модулю 2:



б) арифметическое суммирование по различным модулям:



причем

--- полная арифметическая сумма;

--- арифметическая сумма без учета переноса из старшего разряда;

--- арифметическая сумма с циклическим переносом в младший разряд.

## 71. Синдром.

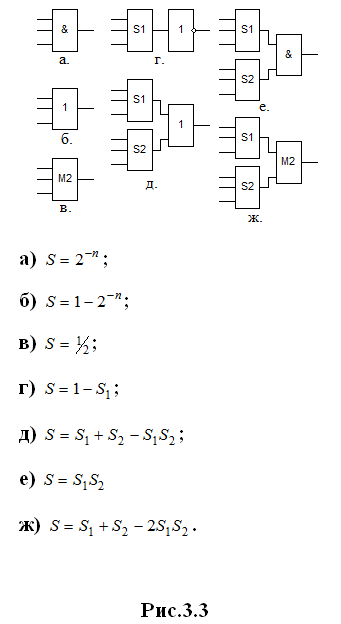
Синдромное тестирование используется при исчерпывающем компактном тестировании. Синдромом булевой функции называется число

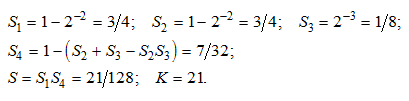


где *К –* число минтермов функции; *n –* число входов проверяемой схемы.

Синдром используется для тестирования комбинационных схем и требует полного перебора входных наборов. Схема называется синдромно-тестируемой, если любая одиночная неисправность меняет синдром.

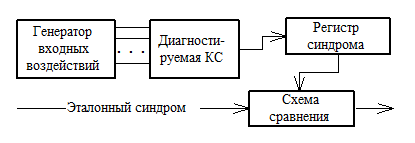
На рис.3.3(а, б, в) приведены синдромы простых *n*-входовых комбинационных схем. При неразветвленных входах соотношение между входными и выходными синдромами схемы, на выходе которой стоит инвертор, схема ИЛИ, схема И или схема сложения по модулю 2, показано на рис.3.3(г,д,е,ж).





Тестовая процедура заключается в подаче на вход схемы всех входных наборов, определении синдрома (обычно с помощью счетчика) и сравнении его с эталонным синдромом (т. е. требуется всего один эталон).

Устройство для тестирования комбинационных схем:



На проверяемую схему подаются от счетчика все входные наборы. Выход проверяемой схемы соединен со входом счетчика синдрома, который подсчитывает число единиц на выходе проверяемой схемы. После перебора всех выходных наборов производится сравнение полученного и эталонного синдромов.

Следует отметить, что единственным различием между синдромом и числом единиц является неявная запятая в регистре (счетчике) синдрома. Если она считается стоящей слева от числа, находящегося в регистре синдрома, то число является синдромом, если справа – то числом единиц.

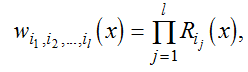
Для уменьшения длины тестов, равной 2n, комбинационная схема разбивается на подсхемы, спроектированные таким образом, что каждая из них проверяется по своему синдрому.

Для реализации синдромного тестирования комбинационные схемы должны проектироваться таким образом, чтобы синдром исправной схемы отличался от неисправной.

## 72. Спектральные коэффициенты

Сжатие информации с помощью спектральных коэффициентов используется при исчерпывающем компактном тестировании комбинационных схем.

Пусть на вход схемы поступает набор x1,x2,...,xn, i1,i2,...,it – номера тех разрядов входного набора функции F(x1,x2,...,xn), зависимость F от которых необходимо определить. Функциями Уолша wi называются функции, принимающие значения ±1 и вычисляемые в соответствии с формулой:



где  - функция Радемахера:



Всего имеется 2n функций Уолша. Например, для функции F(x1,x2,x3) имеется восемь функций Уолша: w0, w1, w2, w12, w3, w13, w23, w123.

**Спектральным коэффициентом** или коэффициентом Уолша называется функция:

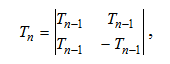


показывающая меру зависимости значения функции от суммы по модулю 2 разрядов xi1 … xil входного набора; S(ij) – зависимость от ij-го разряда.

Коэффициенты Уолша можно вычислить также по следующей фор­муле:

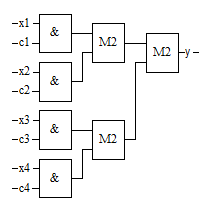


где Тп есть 2nx2n – трансформационная матрица, определяемая следующим образом:



a T0=[l]. Строки матрицы Тп представляют собой значения функций Уолша.

Схема генератора для функций от четы­рех переменных:



Генератор функций Уолша, показанный на этом рисунке, управляет выбором проверяемого спект­рального коэффициента. Накопитель коэффициента представляет собой простой реверсивный счетчик.

Генератор для коэффициентов s0,s1,...,sn тривиален. Для s0 требу­ется, чтобы все константы сi были равны нулю, а для s1,s2,...,sn тре­буется простой мультиплексор.

Для подсчета каждого коэффициента требуется полный прогон тес­товых наборов.

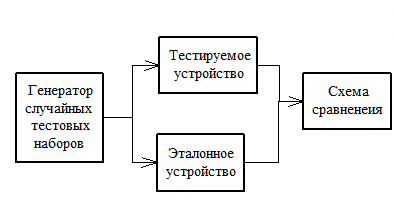
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c1 | c2 | c3 | c4 | y |
| 1 | 0 | 0 | 0 | x1 |
| 0 | 1 | 0 | 0 | x2 |
| 0 | 0 | 1 | 0 | x3 |
| 0 | 0 | 0 | 1 | x4 |
| 1 | 1 | 0 | 0 | x1 xor x2 |
| 1 | 0 | 1 | 0 | x1 xor x3 |
| ... | ... | ... | ... |  |
| 1 | 1 | 1 | 1 | x1 xor x2 xor x3 xor x4 |

Распространена и другая интерпретация спектральных коэффици­ентов, при которой логическому 0(1) ставится в соответствие число –1(+1). Каждый спектральный коэффициент вычисляется умножением значения функции (+1 или -1) на соответствующее значение (+1 или -1) функций Уолша и суммированием произведений по двум наборам входных переменных. При этом значение каждого коэффициента лежит в пределах от —2n до +2n.

## 73. Вероятностное тестирование

**Вероятностное тестирование** характеризуется тем, что на входы проверяемого устройства подаются случайные или псевдослучайные по­следовательности.

Одна из возможных схем вероятностного некомпактного тестиро­вания:



Случайные последовательности подаются на входы проверяемого и эталонного устройств, а выходы обоих уст­ройств сравниваются между собой.

При вероятностном компактном тестировании на входы проверяе­мого устройства подаются случайные тестовые наборы, а результаты тестирования сжимаются одним из способов, приведенных в предыдущем параграфе, и сравниваются с эталонным сжатым результатом.

Вероятностное компактное тестирование (ВКТ) выполняется за два или три шага в зависимости от проверяемой схемы - комбинационной или с памятью. На первом шаге для схем с памятью, носящем название инициализации, на них подается длинная последовательность случайных наборов, цель которой – установка схем в исходное состояние.

Следующие два шага аналогичны для комбинационных схем и схем с памятью: накопление результата и сравнение со сжатым эталоном. Сжатый результат даже в случае неисправности проверяемой схе­мы может не дать точного совпадения с эталоном, а быть лишь доста­точно близким к нему. Причиной этого может быть не зафиксированное началь­ное состояние тестируемых схем с памятью или генераторов случайных наборов, а также вхождение в набор запрещенных входных воздействий. Поэтому при ВКТ проверяемая схема считается исправной, если ре­зультат отличается от эталона на величину, не превышающую погрешность и на­зываемую допустимым отклонением и при кажущейся простоте применения случайных испытательных последовательностей, на практике обычно требуются длительные эксперименты по подбору характера входных воздействий для получения стабильных реакций.

## 74. Общая характеристика методов синтеза детерминированных тестов.

# Задачи