ЭКЗАМЕН ПО АПКС - (надеюсь на печеньки от всех за проделанную работу :3)

1. Краткая характеристика процесса проектирования.

Проектирование начинается с предположений. Результат проектирования - набор документов, содержащих объяснения достаточных для изготовления объекта.

Проектирование – это разработка технической документации изделия, согласно функциональным требованиям. В ТЗ описываются функции, которые должен выполнять объект. После этого описывается назначение, область применения и т.д.

Автоматизация проектирования – это использование ЭВМ при проектировании изделия. Такой способ называется человеко-механическим.

Автоматическое проектирования – это проектирование без участия (использования) людей.

Автоматическое проектирование не используется, потому что технология производства ЭВМ очень быстро развивается.

2. Определение САПР.

Система автоматического проектирования (САПР) - это система, которая обеспечивает проектирование объекта без участия человека и содержит в себе различные методы и средства для осуществеления автоматического проектирования. Данные методы и средства также называются обеспечениями, и бывают следующих типов: Техническое, Математическое, Программное, Лингвистическое, Информационное, Методическое, Организационное.

3. Виды обеспечений САПР.

Техническое - состоит из технических связей (ЭВМ, ПК, сеть, ВУ и прочее).

Математическое - модели объекта, частей объекта, методы выполнения объектных процедур. Есть инвариантные (для решения любых задач) и объектно-ориентированные (учитывают спецификацию конкретной задачи).

Программное - Программы бывают общесистемные и разработанные специально для проектирования (бывают базовые управляющие и прикладные)

Лингвистическое - все языки, с помощью которых можно решить данную задачу. Как пример, язык программирования. Языки бывают универсальные и объектно-ориетированные (специальные под конкретную задачу).

Информационное - база данных и база знаний.

Методическое - методика выполнения проектирования (документы, правила...)

Организационное - регламентирует организационную структуру системы для автоматического проектирования. Документы и всякое такое.

4. Принципы системного подхода к процессу проектирования.

При данном подходе система рассматривается как взаимодействующиек элементы причём на разных уровнях жизненного цикла. Жизненный цикл - составление технического задания, пректирование, изготовление... Принципы системного подхода:

СТРУКТУРНЫЙ - Есть множество элементов из которых надо создать систему. Надо, чтобы выполнилось техническое задание.

БЛОЧНО-ИЕРАРХИЧЕСКИЙ - У нас есть некая система выполняющая функцию F. На вход подаём X, получаем Y на выходе. Функцию можно разделить на мелкие функции, совокупность которых выполняет целевую функцию: F = {f1,f2,f3,...,fn,...}. Каждую функцию можно реализовать и также разделить на более мелкие функции... Декомпозиция идёт вплоть до элементарных.

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ - при разработке программ.

5. Общий подход к делению проектирования.

Этапы проектирования:

1. *Деление проектирования по времени вып. работы.*

- Научно-исследовательские работы.
- Опытно-конструкторские работы (Эскизное проектирование).
- Рабочее проектирование.
- Создание и испытание опытного образца.

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

- ФУНКЦИОНАЛЬНОЕ ПР - НИЕ (распределение функций по техническим и программным свойствам. Структурные и функциональные схемы).
- АЛГОРИТМИЧЕСКОЕ ПР - НИЕ (по программным свойствам, разрабатываются алгоритмы и прочее...)
- КОНСТРУКТОРСКОЕ ПР - НИЕ (физическая организация схем, прицинпиальные схемы, размещение на плате, полная техническая документация).
- ТЕХНОЛОГИЧЕСКОЕ ПР - НИЕ (разработка технологий для создания устройства).

3. *Горизонтальное.(по смежности проектных подразделений)*

Каждый из этапов вертикального деления делятся, также по горизонтали:

ФУНКЦИОНАЛЬНОЕ ПР - НИЕ

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

АЛГОРИТМИЧЕСКОЕ ПР - НИЕ

1. Разработка алгоритмов
2. Разработка программ
3. Разработка микропрограмм
4. Разработка ОС.

КОНСТРУКТОРСКОЕ ПР - НИЕ

1. Проектирование "шкаф-стойка";
2. Проектирование панелей
3. Проектирование модуль, кристалл, ячейка.

ТЕХНОЛОГИЧЕСКОЕ ПР - НИЕ

1. Маршруты 2. Карты 3. Операции 4. Оснастки.

6. Деление 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 специализированные САПР. Результаты исследований не являются обязательными для испытаний.
- Опытно-конструкторские работы (ОКР)
  Формируется и согласовывается ТЗ на разработку нового изделия и выполняются ОКР. Прорабатываются возможные варианты проекта, можно ли реализовать данное ТЗ, задаётся элементная база, нет ли необходимости ее разработать. Синтезируется эскиз проекта. Результат – эскиз будущего устройства.
- Техническое (рабочее) проектирование
  Основной этап Проектирования, на котором непосредственно разрабатывается проект. Результат – документация на изготовление объекта (принц. схемы со всеми описаниями)
- Производство опытного образца
  Испытания опытного образца. По результатам возможны варианты.

7. Деление процесса проектирования по характеру выполняемых работ.

ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ. Исходные данные – ТЗ и промежуточные результаты алгоритмического проектирования. Цель – разработка функциональных, принц., структурных схем. Уточняется ТЗ, распределяются ф-ии объекта, кот будут выполняться при помощи технических средств.

АЛГОРИТМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ – Разрабатываются алгоритмы реализации функций объекта, разрабатываются программы и микропрограммы для объекта.

КОНСТРУКТОРСКОЕ ПРОЕКТИРОВАНИЕ – разрабатываются конструкции(железо), которые будут реализовывать данный проект.

ТЕХНОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ – разработка технологии изготовления данного продукта, определяются маршрутные карты, инструменты для реализации и прочее.

8. Деление процесса проектирования по блочно-иерархическому методу.

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

- Функциональное

1. Системный уровень – определяется общая структура проекта; определяется функц. и структурная схема, после построения схемы определяется нужно ли проектировать новые элементы. Если да, то 2
2. Функц.-логический – определение функциональные и логические элементы будущего устройства, в частности монтаж ; определяется заполнение платы реальными элементами; компоновка и размещение элементов в модулях; трассировка печатной платы, анализируется электрические параметры будущего изделия.
3. Технический
4. Компонентный

- Алгоритмическое

1. Программирование алгоритмов.
2. Программирование системы
3. Программирование Модулей
4. Проектирование мета-программ. – программисты

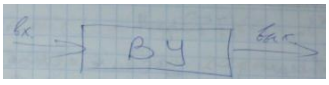
- Конструкторское

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

- Техническое – принц. схема, маршрутные карты, операции.

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.



С каждым этапом все более высший уровень детализации.

Процесс блочно-иерархического проэктирования может быть **нисходящий** и **восходящий.**

«Линии отрыва» между уровнями определяются математическим аппаратом, который может быть использован при проектировании на данном этапе.

1) На самом верхнем уровне используется теория вычислительных систем для решения задачи синтеза и имитационное моделирование для анализа.
2) Синтез – ПТЦА. Анализ – логическое моделирование. Получаем функц. и логические схемы.
3) Синтез и анализ – теория графов, теория множеств.

10. Типовые проектные процедуры

Проектные процедуры - часть этапа, выполнение которого запоминается получением проектного решения. Оформление расчёт узла, нечто такое...

Анализ (функции и параметры из структуры), синтез (структура из функций и параметров).

Основными проектными процедурами являются синтез и анализ. Синтез - непосредственная разработка проекта, получение проектных вариантов. Синтез бывает структурный (определение структуры, топологии, состава элемента объекта) и параметрический (различные характеристики и параметры объекта). Параметрический синтез по сути является задачей оптимизации, если целью выбирается выбор лучших характеристик объекта, или выбором синтеза наилучшего решения данной задач по различным критериям. Оптимизация в свою очередь может быть также параметрической или структурной.

Стуктурный можно определить задачу математически.

O = {F, S, P} , O - описание объекта, F - функции, S - структура, P - параметры.

Структурный синтез - преобразование функционального и параметрического описания в структуру и параметрическое описание элементов. {Fc , Pc } -> { Sэ, Pэ}

Анализ - задач исследования проэктируемого объекта. Модели могут быть физические или математические, которые в свою очередь бывают функциональные и структурные. Процедуры анализа бывают одновариантные или многовариантые. В первом случае это значит что при внешних и внутренних параметрах известных, нужны выходные результаты, во втором - исследования происходят в некоторой области пространства внутренних параметров. Т.е это возможно многократное выполнение одновариантного анализа.

11. Задача синтеза в процессе проектирования

Синтез - создание/проектирвание описание объекта, выполняющего определённые функции и содержащего определённые параметры. Задача синтеза выполняется в выбранном множестве элементов из которых можно составить объект, или которые могут реализовать определённое множество функций/объектов.

Исходные данные для синтеза: описание функций и параметров объекта а также некоторые ограничения на значение параметров. Результатов является некотарая структура содержащая данные параметры и заданный класс функций.

Структура - множество S = {C, H}, где C - множество элементов, H - множество связей между элементами.

Две структуры равны есливыполняется следующее утверждение ({F1} == {F2}), ({C1} == {C2}), ({H1} == {H2}).

Две структуры эквивалентные, елы выполняется следующее ({F1} == {F2}), ({C1}<>{C2}) и/или ({H1}<>{H2}).

Синтез может иметь ормальные методы решения - тогда задача автоматизации синтеза алгоритмически резрешима, иначе могут использоватся только ручное решение или применение эвристических алгоритмов (полный перебор).

Синтез бывает структурные (определение структуры объекта, связей, состава элементов) и параметрический (числовые характеристики объекта).

Стуктурный можно определить задачу математически.

O = {F, S, P} , O - описание объекта, F - функции, S - структура, P - параметры.

Структурный синтез - преобразование функционального и параметрического описания в структуру и параметрическое описание элементов. {Fc , Pc } -> { Sэ, Pэ}

Параметрический синтез по сути является задачей оптимизации, если целью выбирается выбор лучших характеристик объекта, или выбором синтеза наилучшего решения данной задач по различным критериям. Оптимизация в свою очередь может быть также параметрической или структурной.

12. Задача анализа в процессе проектирования

Анализ - необязательный в процессе проектирования но всё же необходимый. Это изучение свойств результирующего проектируемого объекта, тоесть определение функций и параметров из имеющей структуры. Предметом иследования является исследование свойств F, S, P-элементов описание иверификация их качество проектирования. Задача анализ в отличие от синтеза всегда разрешима. Задача анализа решается с помощью моделирования. Модели бывают физические и математические, которые в свою очередь бывают структурные и функциональные. Бывают одновариантный анализ (определение некоторых выходных параметров системы исходя из имеющихся внутренних и внешних параметров) и многовариантный (задача поиска в некоторой области, что по сути является множественным выполнением одновариантного анализа).

13. Задача оптимизации в процессе проектирования

Задача оптимизации играет очень важную роль при решении задачи синтеза. Задача оптимизации может быть структурной, когда выполняется определение оптимальной структуры объекта, и параметрической, когда идёт поиск оптимальных значений параметров проектируемого объекта.

Оптимальными считаются те значения, которые удовлетворяются ТЗ и являются наилучшими из достижимых.Задача оптимизации подразуевается превращение структурно-параметрической модели объекта в математическое описание экстремальной задачи. Экстремальные задачи (цель оптимизации) работают с учётом критериев оптимизации.

Критерии это предпочтения оптимальных результатов которые могут быть. Основой критериев является целевая функция F(x), где x - вектор параметров и характеристик оптимального объекта. Множество векторов x является вектором различных вариантов объекта. По целевой функции можно определить степеть оптимальности объекта

С другой стороны есть ещё один важный критерий оптимизации, это результирующая функция параметров объекта. Это значит что лучшим выбирается тот объект у которого F(X) меньше или больше всего, где X - некий набор характеристик.

Помимо целевой функции и перечня минимаксимализации параметров в процедурах оптимизации также применяются ограничающие функции f(x) = 0, f(x) <>0, частным случаем которого является выбор отрезка параметров ai<=xi<=bi, где xi некоторый параметр объекта, а ai и bi - ограничения накладываемые на этот параметр. Данная область называется допустимой областью параметра. Если в объекте все параметры находятся в области данного отрезка, то он называется оптимальным, если некоторые из параметров вне этого отрезка, он называется квазиоптимальным.

Допустимая область Xd эт область пространства управляемых параметров, в которой выполняются заданные ограничения. В сумме, задача оптимизации при проектировании имеет вид: экстремизировать целевую функцию F(X) в области Xd, заданной ограничениями H(X) и ф(X).

В таком случае задача оптимизации стаёт задачей математического программирования. Если F(X), H(X) и ф(X) линейные - это задача линейного программирования, если одно из них нелинейно, то нелинейного. Если множество X - ддискретны, то это задача дискретного программирования, если Xd - это пространство булевых переменных, то это задача бивалентного программирования.

14. Математическая постановка задачи оптимизации.

Оптимальный - удовлетворяющий целевую функцию и вкладывающийся в имеющиеся ресурсы. Основа задачи оптимизации - поиск оптимального значения по заданным критериям. Основа критерия - целевая функция F(x), где x - множество управляющих параметров. Фиксация значений вектора параметров представляет некоторое решение задачи оптимизации, таким образом, на некоторые из параметров могут накладыватся ограничения, а на некоторые нет. Ограничения могут быть строгими, задаватся математически в виде равенств или неравенств, или прямыми ограничениями (некоторой областью). Область параметров, которые удовлетворяют области ограничений называется допустимой обастью Xd.

экстремум f(x) должен находится в области допустимых значений параметров

$$\underset{x \in xD}{extr}\, F(x),\ где\ \text{xD}=\{\text{x}|,\ \varphi(\text{x})=0,\ \psi(\text{x}) \le 0$$

таким образом задача оптимизации это задача математического программирования.

если функции линейны -линейное программирование.

если некоторые из них нелинейны - нелинейное программирование

если параметры дискретны (xeZ) - дискретное программирование.

если параметры булевые (xe{0,1}) - бивалентное программирование.

15. Общая характеристика критериев оптимизации.

Критерий оптимальности – это правило предпочтений сравниваемых вариантов.
Если во время проектирования выбирается параметр, который наиболее полно описывает объект, и ему отдаётся безусловное предпочтение, то он выбирается за целевую функцию, такое правило описывает основу ***частных критериев.***
Множество параметров на которые накладываются ограничения вычисляются при оптимизации их целевой функции, а те на которые не накладываются ограничения выбираются такими какими они в результате оптимизации вышли. Такие задачи называются однокритериальными и они довольно просты, тоесть существуют математические средства для их разрешения.
Задачи проектирования основанные на множестве критериев оптимизации наз. многокритериальными или задачами векторной оптимизации. Тоесть есть один некий составной критерий, который состоит из нескольких частных критериев F(X) = Ф(F1(X),F2(X), F3(X),...,Fn(X)), который затем максимизируется или минимизируется. Таким образом объединения частных критериев в составной выполняется с помощью существующих алгоритмов составления. Среди них есть аддитивный, мультипликативный и минимаксный (максминный). Критерии могут быть детерминированными или статистическими (если учитывается вероятность разброса параметров).

16. Аддитивный критерий оптимизации

Составной критерий оптимизации получается путём сложения нормированных значений частных критериев. Нормированные критерии - это отношение реального значения частного критерия к нормирующему значению. Есть три подхода к выбору нормирующего значения:

1. Нормирующее значение заданное заказчиком, подразумевается что оно уже есть в ТЗ и заданные значения критериев воспринимаются как образцовые.
2. Выбор в качестве нормирующих, максимальные значения частных критериев, определённые в области существования проектных решений.
3. Используется разность между минимальным и максимальным значением в области компромисса.

Выбор подхода является субъективным и должно детально обосновывается.

ФОРМУЛА ВЫЧИСЛЕНИЯ АДДИТИВНОГО КРИТЕРИЯ ОПТИМИЗАЦИИ

$$F(X) = \sum_{i=1}^{n} C_i \frac{F_i(X)}{F_i^{(0)}(X)}$$

где C_i - весовой коэффициент i-того критерия

$F_i(X)$ - значение частного i-того критерия

$F_i^{(0)}(X)$ - нормирующее значение i-того критерия

Позитив: нет логических проблем, связанных с установлением логических взаимосвязей между частными критериями различной размерности.
Недостатки: формальный математический приём упрощающий задачу до того момента когда её можно решить. Также может быть взаимная компенсация некоторых критериев.

17. Мультипликативный критерий оптимизации

Используется принцип справедливой компенсации абсолютных значений нормированных частных критериев. В целом ряде задач используется не абсолютное значение, а принцип относительной компенсации, то есть справедливым следует считать такой компонент, когда суммарный уровень относительного снижения значений одного или нескольких критериев не превышает суммарного уровня относительного увеличения значений других критериев.

$\sum_{i=1}^{n} \frac{\Delta F_i(x)}{F_i(x)} = 0$, где n – количество параметров, $F_i(x)$ – начальное значение частного коэффициента, $\Delta F_i(x)$ –изменение значений в новом варианте.

С учетом коэффициента значимости $\sum_{i=1}^{n} C_i \frac{\Delta F_i(x)}{F_i(x)} = 0$

С учетом $\Delta F_i(x)$ $f_i(x)$, то формулу можно преобразовать

$$\sum_{i=1}^{n} \frac{\Delta F_i(x)}{F_i(x)} = \sum_{i=1}^{n} d\left(\ln F_i(x)\right) = d\left(\ln \prod_{i=1}^{n} F_i(x)\right) = 0$$

Это всё сложно реализовать, поэтому необходимо варьировать параметрами в произведении. $F_i(x) = \prod_{i=1}^{n} F_i(x)$ – критерий является мультипликативным.

С учетом коэффициента значимости используют $F_i(x) = \prod_{i=1}^{n} C_i F_i(x)$

ФОРМУЛА ВЫЧИСЛЕНИЯ МУЛЬТИПЛИКАТИВНОГО КРИТЕРИЯ ОПТИМИЗАЦИИ

$$F(X) = \prod_{i=1}^{n} F_i^{Ci}(X) = \prod_{i=1}^{n} C_i F_i(X)$$

где C_i - весовой коэффициент i-того критерия.

$F_i(X)$ - значение частного i-того критерия.

$F_i^{Ci}(X)$ - значение частного i-того критерия с учётом его коэффициента.

Достоинства: При использовании данного критерия можно не использовать нормированные значения.

Недостатки: Компенсирует недостаточную величину частного критерия избыточной величиной другого, тоесть он сглаживает все уровни критериев.

18. Минимаксные критерии оптимизации

Сущность данных критериев в том, что при проектировании реально сложных систем зачастую трудно или невозможно определить зависимость между критериями, потому что их реально очень много.Поэтому стараются найти такое множество переменных проектирования, при котором нормированные значения всех частных критериев становятся равными между собой, т.е.

$$f_i(x) = k, i = \overline{1,n}$$

С учётом весовых коэффициентов важности C_i

$$c_i f_i(x) = k, i = \overline{1,n}$$

Таких штук много, поэтому прицнип минмаксности используется. Все минимальные посчитанные нормированные критерии какбы подтягиваются , они ведь оказались в исходном решении наименьшими.

Формально принцип формулируется следующим образом:

нужно выбрать $X^{(0)} eX$, на котором реализуется максимум из минимальных, тоесть $F(X^{(0)})$=max min $\{ f_i(x) \}$ (этот принцип иногда называют принципом гарантированного результата).

Если частные критерии надо минимизировать, то это минмаксная задача, тоесть $F(X^{(0)})$= min max $\{ f_i(x) \}$.

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.

В многокритериальных задачах очень важно проводить некую ранжировку частных критериев по важности. Осуществляют ранжировку заданных критериев с помощью весового коэффициента C_i который численно отображает важность данного критерия. Значения весового коэффициента выбираются исходя исходя из анализа текущего технологического уровня мира, из требований системы и из существующих возможностей реализации. Есть несколько методов по определению весового коэффициента.

Метод ранжировки. Есть l (эль) экспертов которым просят выставить критерию некоторую важность (ранг). Каждый из экспертов сортирует критерии по рангам от 1 до n (при этом 1 ранг - это число n получается, а n - 1). Таким образом весовые коэффициенты вычисляются по формуле (при учёте что r_i^k - ранг k-того специалиста для i-того критерия):

$$C_i = \frac{\sum_{k=1}^{l} r_i^k}{\sum_{i=1}^{n}\sum_{k=1}^{l} r_i^k}, i = \overline{1,n}$$

Метод приписывания баллов. Эксперты оценивают по шкале 0-10 (при этом можно устанавливать дробные и даже одинаковые значение). Тогда r_i^k заменяется H_i^k и он равен

$$H_i^k = \frac{h_i^k}{\sum_{k=1}^{l} h_i^k}, i = \overline{1,n} \text{ ===>>> } C_i = \frac{\sum_{k=1}^{l} H_i^k}{\sum_{i=1}^{n}\sum_{k=1}^{l} H_i^k}, i = \overline{1,n}$$

21. Общая характеристика системного уровня проектирования

Главная особенность системного этапа проектирования – это то, что задачи, решаемые здесь, трудно поддаются формализации. Обычно на этапе системного проектирования решаются следующие задачи:

1. Определение принципов организации ВС;
2. Выбор архитектуры;
3. Построение структурных схем.

Эти задачи плохо формализуются, особенно в части синтеза. Поэтому сначала выполняется с помощью человека-проектировщика синтез первоначальных структур. А их корректировка выполняется на основе автоматизированного анализа (формальные методики которого существуют). Задача ***синтеза*** решается с помощью теории сложности систем и теории вычислительных систем. ***Анализ*** использует Теормию Массового Обслуживания.

22. Задача синтеза и анализа на системном уровне проектирования

Системный уровень проектирования это самый высокий уровень. Он может и не выполнятся если уже есть структура. Задача синтеза решается с помощью теории сложности систем и теории вычислительных систем. В основном синтез производится вручную с использованием ЭВМ для осуществления формальных методик. Задача анализа имеет формальные методики.На этом этапе нужно определить устройства и связи между ними. Для характеристики устройств используют вероятностные величины. Анализ использует Теормию Массового Обслуживания. На выходе получаем структуру (схему). В данном случае интересует оборудование, а не его характеристики.

23. Общая характеристика СМО

Система массового обслуживания - система в которой обслуживаются заявки. Оно состоит из стационарных объектов (обслуживающие аппараты) и из хранилищ (память). Обслуживающие устройства - могут быть заняты или свободны и к каждому ресурсу очередь одна или их несколько. Хранилище характеризируется общим и занятым объёмом. Если объём памяти свободен для заявки, то она записывается в память.
Временные объекты: заявки. Характеристика их: обслуживается/ожидает.
Если у заявки естьь приоритет, то система называется с приоритетом, если без, то они обслуживаются в порядке поступления.
Существуют определённые принципы по которым заявки отбираются из очереди - дисциплины обслуживания (FIFO, LIFO, RR и прочее).Приоритеты также бывают 3-х типов: Абсолютные, Относительные, Динамические. 1 - если пришла заявка с большим приоритетом, то она прерывает выполнение меньшей, 2 - нет прерывания, 3 - приоритет меняется во время работы.
СМО исследуются двумя способами: аналитически и имитационно.

24. Аналитическая модель СМО

Аналитическая модель - описание всех временных характеристик, если можно получить, то такая модель предпочтительнее имитационной. Аналитические модели используются для предварительной оценке качества проектов при значительных допущениях. Используется для простейших СМО. Основные свойства: безприоритетное, FIFO, времена обслуживания с экспоненциальным законом распределения и заявки аппроксимируются простейшим потоком, стационарность, ординарность, отсутствие последействия.
(*Стационарность* – вероятность поступления заявки одинакова в любой момент времени,
Ординарность – невозможность поступления более одной заявки в любой момент времени,
Отсутствие последствия – вероятности поступления двух и более заявок не зависят друг от друга)
Аналитическая модель характеризируется конечным множеством состояний - это конечные марковские процессы.
НА ВХОДЕ: матрица вероятностей переходов и начальные условия. т.е граф (вершины - состояния, дуги - вероятности перехода и интенсивность перехода). Информация об установлении состояния имеет вероятностный характер, и за основу взято уравнения Калмагорова:

$$P_i(t) = \sum_{jeJ} Pij(t1) * Pi(t) - \sum_{keK} Pik(t1) * Pi(t)$$

В стационарном состоянии производная от pi по t = 0/
НА ВЫХОДЕ: Нужно получить аналитическую зависимость, среднее число заявок, ожидание в очереди к устройству, среднее время нахождения заявок в системе, среднее время ожидания заявки в очереде.

25. Имитационная модель СМО

Это модель описывающая функционирование системы. Так как модель - СМО, то должны быть записаны все свойства. Все значения, описывающее функционирование системы - случайные величины.
Модель потока заявок - алгоритм, вырабатывающий величину по заданному закону распределения, определяющий или имитирующий время поступления заявки (значние промежутка времени между двумя значениями).
Источники заявок бывают зависимые и независимые.
Зависимые - определяют время появления новых заявок в зависимости от поведения другой заявки, которая наз. синхронизирующей.
Модель устройства обслуживания - алгоритм, выработки случайного числа по заданному закону расспределения имитирующего работу устройства (обслуживание заявки).Связи определяются программно и являются отдельным аппаратом моделирования. Бывают нескольких типов: 1 - служат для направления заявки в зависимости от типа заявки и выполнения некоторых условий. 2 - для разделения и объединения заявок, 3 - для изменения параметров заявок.Время дискретное и изменяется только после того, когда завершились все действия относящиеся к данному моменту времени. Все события происходят мгновенно. Во время моделирования идёт накопления всех данных всех параметров. Движением заявок управляет программа. Сам маршрут заранее дан или определён по типу заявки. Имитация происходит во времени. Время изменяется ti = ti-1 + dt. dt - всегда переменная величина, и каждый такой прыжок выполняются все события сразу.

26. Динамическая структура моделирования СМО

Динам. структура объекта представлена понятиями: 1) события 2)активности 3) процессы

Здесь необходимо время Т : 1)Время реальное

2)Время. машинное – реальное время вып. программ.

3)Системное или модельное время – модель системного времени в машине.

Активность – любое действ. в вашей системе. Активности представлены промежутком времени, необходимым для вып. данного действия. { t1 hold (t); wait(t)}. Активности обязательно заданы промежутком времени.

Всегда необходимо указывать время выполнения данной активности, вычисляем время завершения данной активности.

Событие- любое изм. состояния системы. События не требует времени, они происходят мгновенно. События бывают:

1) Следования – предполагают последовательность выполнения некот. активностей.
2) Изменения состояния – предполагает изменения состояния системы.

Процесс – логически связанный набор активностей. Любой процесс может быть представлен активностью и наоборот. Процессы объединяют множество промежутков времени.

Процесс является динамическим объектом, представляющим единый акт выполнения совокупности логически связанных процессов. Выполнение процесса происходит во времени. В каждый момент времени в системе может выполнятся несколько процессов одного класса, находящихся в разных стадиях выполнения.

Любая система моделир.- последовательность событий. Между этими событиями происходят действия – активности. Эти активности тоже можно расписать как последовательность событий. Одни и те же действия можно представ. либо в виде активностей, либо в виде процессов.

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. Событийный метод может существенно ускорить моделирование по сравнению с пошаговым методом, в котором на каждом такте анализируются состояния всех элементов модели.

Моделирование начинается с просмотра операторов генерирования заявок, т.е. с обращения к моделям источников входных потоков. 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 имитационной модели. В СБС события упорядочиваются по увеличению 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 имитационной модели заявки А, с которой связано моделируемое событие. Пусть этим местом является устройство Х. Далее программа моделирования выполняет следующие действия (рис. 1.7):

1) изменяет параметры состояния устройства Х исходя из заявки (она может менять)
2) прогнозируется время наступления следующего события и расчитываются новые которые могла вызвать предыдущая заявка (и заносятся в СБС)
3) происходит имитация движения заявки А в сетевой имитационной модели (СИМ) по маршруту, определяемому заданной программой моделирования (или она задерживается в очереди)
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 программирования, ориентированные на описание событий.

Языки: GASP, SIMSCRIPT, SMPL.

События объединяют в классы событий. Каждый класс описывается набором конструкций (опер.), характерных для вып. данного события.

Наличие такого опис. события , которое примен. в процессе моделирования определенное количество раз, говорит о последовательности событий одного класса. Каждое событие одного класса наз. динамическим экземпляром описания. События принадлежащие одному классу хар-ся одним и тем же набором операций, выполняемых над различными объектами одного класса, при чем принадлежность объекта данному классу каждый раз определяется (напр. значением ссылочных переменных либо динамических , по контексту). В этих языках все события задаются и описываются явно. Класс события задает набор действий, вызв. заявками.

ЭМУЛЯЦИЯ: после генерации всех типов заявок формируется одна, которая будет обрабатыватся. Заявки заносятся в список событий (заносится имя заявки, время и событие, которое произойдёт). Из списка будущих событий выбирается ближайшее к текущему времени. Изменяется состояние счётчика времени. Выполняются все действия, относящиеся к данному событию и планируется время следующего события. Операция повторяется снова.

29. Языки программирования, ориентированные на описание процессов.

Языки: SOL, ASPOL.

Процесс является динамическим объектом, представляющим единый акт выполнения совокупности логически связанных процессов. Выполнение процесса происходит во времени. В каждый момент времени в системе может выполнятся несколько процессов одного класса, находящихся в разных стадиях выполнения. Каждый класс процесса описывается совокупностью конструкций языка, представляющих выполнение данного процесса. Программа ориентир. на описание событий длинная, её можно сократить заменив на описание процессов (будут описываться только события следования). Каждый процесс инициируется другим процессом, который может находится как внутри системы, так и нет. Конструкция описывающая поведение класса процессов, состоящая из указания активностей, входящих в процесс, в определенном соотношении следования условий управляющих их выполнение и воздействий, оказываемых процессом на атрибуты и состояния активных и пассивных объектов системы.

30. Краткое описание ASPOL

Программа в ASPOL состоит из основной программы, которая наз. SIM<имя>, это прогр. опис. глобальные переменные, устр-ва памяти и инициирует выполнение первых активностей и определяет время моделирования. Для определения этого времени определяют кол-во заявок а заранее задается время моделирования. Основной процесс выполняет управления всеми остальными процессами.

Любой процесс начинается: **process**< имя> (лок. перменные)

заканчивается end process<имя>

Для инициирования выполнения процесса надо в другом процессе инициализ. этот процесс и фактические переменные: **initiate** <имя> (переменные)

Каждый процесс (и экземпляр класса) может иметь свой приоритет (неявно он равен приоритету процесса инициатора). Для изм. приоритета исп. конструкцию priority =<выражение>

Любой процесс может находится в одном из 4х состояний:

1)Готовности – процесс уже инициирован, но ещё не выполняется.

2)Выполнения

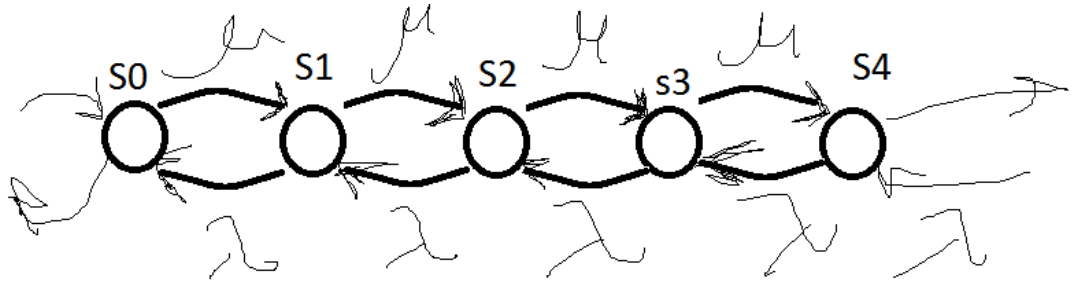
3)Задержки (hold (t)

4)Ожидания (ожидание освобождения устройства)

31. Краткое описание GPRS

Язык GPSS (General Purpose Simulation System), ориентированный на процессы.
Модель (программа) на языке GPSS представляет собой последовательность операторов (их называют блоками), отображающих события, происходящие в СМО при перемещениях транзактов. В интерпретаторах GPSS реализуется событийный метод.
Модель в GPSS строится из отдельных элементов, называемых объектами. Имеется всего четыре вида объектов:
а) Динамические, или транзакты;
б) Статические, или оборудование;
в) Статистические;
г) Операционные блоки.
Состояние модели в любой момент времени определяется совокупностью состояний всех объектов, составляющих модель; смена состояний модели предполагает изменение состояния хотя бы одного объекта. Состояние модели изменяется лишь тогда, когда динамический объект - транзакт - проходит через операционный блок.

32. Пример аналитической модели СМО



	S0	S1	S2	S3	S4
S0	-λ	λ			
S1	μ	-μ-λ	λ		
S2	0	μ	-μ-λ	λ	0
S3			μ	-μ-λ	λ
S4				μ	-μ-λ

Необходимо составить модель уравнение Колмогорова

$$-\lambda * p0 + \mu * p1 = 0$$

$$\lambda * p0 - (\mu + \lambda) * p1 + \mu * p2 = 0$$

$$\lambda * p1 - (\mu + \lambda) * p2 + \mu * p3 = 0$$

$$\lambda * p2 - (\mu + \lambda) * p3 + \mu * p4 = 0$$

Выражаем уравнения через P0 - делим на μ

$$p1 = \frac{\lambda}{\mu} * p0 = a * p0$$

$$p2 = \frac{\mu + \lambda}{\mu} * p1 - \frac{\lambda}{\mu} * p0 = (1 - a)p1 - a * p0 = a^2 p0$$

$$\sum_{i=1}^{\infty} pi = 1$$

$$p0 = \frac{1}{1 + a + a^2 + a^3 + \ldots} = 1 - a;$$

$$Nar = \sum_{i=1}^{\infty} pk * k = p1 - 2p2 + 3p3 + \ldots + kpk. = a(1 - a)(1 + 2a + 3a^2 + \ldots) = \frac{a}{1 - a}$$

$$Qav = p2 + 2p3 + 3p4 + \ldots = \frac{a^2}{1 - a}; Nav = \lambda * Tav; Qor = \lambda * Tor;$$

$$Tav = \frac{a}{\lambda(1 - a)} = \frac{1}{\mu - \lambda}; Tor = \frac{a^2}{\lambda(1 - a)} = \frac{a}{\mu - \lambda}$$

Теперь, меняя μ и λ мы можем получить значения характеристик для различных систем.

33. Пример имитационной модели СМО

Транспорт 1 с грузом отправился из пункта A в пункт C через пункт B. Одновременно из пункта D в пункт E через пункт B отправился транспорт 2. Скорости движения транспортов распределены по нормальному закону с математическими ожиданиями V_1 и V_2 и стандартными отклонениями σ_1 и σ_2 . Построить алгоритм имитационной модели (ИМ) с целью определения вероятности встречи транспортов 1 и 2 в пункте B. Расстояние от пункта A до пункта B S_1 , а от пункта D до пункта B - S_2. Событие встречи считать состоявшимся, если их времена прибытия в пункт B либо равны, либо отличаются на величину, не превышающую Δt.

Возьмем две последовательности нормально распределенных случайных чисел:

$V_{11}, V_{12}, \ldots , V_{1i}, \ldots, V_{1N}$;

$V_{21}, V_{22}, \ldots, V_{2i}, \ldots, V_{2N}$;

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

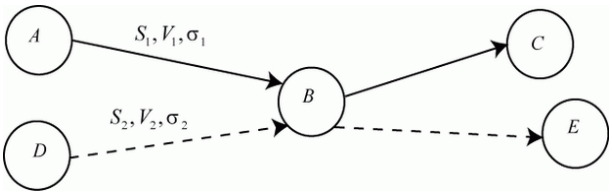


Схема движения транспортов

1. Имитируем движение транспортов 1 и 2 до пункта B со скоростями V_{11} и V_{21} соответственно, взятыми из последовательностей нормально распределенных случайных чисел.

2. Вычислим время t_1 и t_2 прибытия в пункт B транспортов 1 и 2 соответственно:

$$t_1 = \frac{S_1}{V_{11}}, t_2 = \frac{S_2}{V_{21}}$$

3.Оценим результат имитации движения транспортов 1 и 2, т. е. установим факт наличия или отсутствия их встречи:

3.1 если $|t_1 - t_2| \le \Delta t$, встреча состоялась;

3.2 если $|t_1 - t_2| > \Delta t$, встреча не состоялась.

4.Если встреча состоялась, зафиксируем этот факт увеличением значения M на 1, т. е. M = M + 1(вначале M = 0).

5.Для нахождения вероятности встречи транспортов 1 и 2 повторим имитацию их движения N раз.

Рассчитаем вероятность встречи: $P(\Delta t \le |t_1 - t_2|) = \frac{M}{N}$

34. Общая характеристика уровня функционально-логического проектирования.

На этом уровне выполняется составление функциональных и принципиальных схем устройств. Входными данными для этого уровня являются результаты проектирования на системном уровне, тоесть уже известны наборы устройств, их характеристики, какую функцию они выполняют и структура системы (возможно особенности реализации, набор команд и прочее...). Тут уже происходит переход от структурного/системного уровня к более детализированному уровню, который позволяет реализовать структурные компоненты с помощью простейших элементов (например вентили И, ИЛИ, НЕ).

На этом уровне как и на системном решаются задачи синтеза и анализа. Синтез решается с помощью теории цифровых автоматов, а анализ решается с помошью методов логического моделирования.

35. Задача синтеза на уровне функционально-логического проектирования.

Задача синтеза решается с помощью теории цифровых автоматов. Во время решения данной задачи используется блочно-иерархический подход. Тоесть система разбивается и проэктируется на более мелком/нижнем уровне, до тех пор, пока элементы составляющие структуру системы не будут простейшими. Например такие элементы могут быть следующего типа: комбинационные схемы, блоки с памятью, функциональные узлы, узлы с памятью, нестандартные блоки (без формального описания), генераторы. Комбинационные схемы обычно отображаются в виде системы/таблицы булевых функций (yi=f(x1,x2,...,xn)) и автоматизация проектирования таких схем не предоставляет сложности. Блоки с памятью отображаются как конечный автомат S = {A,X,Y,δ,λ,α0}, где A - множество внутренних состояний, X - множество внешних сигналов, Y - множество выходных сигналов, δ - функции переходов, λ - функция выходов, α0 - начальное состояние. Синтез решается так же, ибо существует формализация процесса автоматизации. К функциональным узлам относятся сумматоры, мультиплексоры и прочее, автоматизировать такие структуры сложнее, поэтому они имеют диалоговый смысл (общение разрабтчика с машиной). То же самое и с узлами с памятью и с нестандартными блоками и с генераторами.

36. Задача анализа на уровне функционально-логического проектирования.

Чаще всего после синтеза и составления структуры состоящей из простейших элементов (вентили И, ИЛИ, НЕ и прочее...) необходимо произвести анализ, чтобы удостоверится в правильности работы разработанной схемы/системы. Для этого применяют логического моделирование. Задача анализа необходима потому что большинство этапов синтеза выполняет человек лично. Цель моделирования в том, чтобы эмулировать работу схемы, без её физического построения. Задача анализа схем состоит из двух задач: статический анализ, динамический анализ.

Статический проверяет корректность этапа синтеза:

- установления функции которую выполняет схема, определение характеристик, входных сигналов, проверка корректности связей внутри схемы и прочее.

Динамический, в дополнение к статическому, проверяет также переходные процессы, происходящие внутри схемы. это могут быть частотные характеристики, измерение параметров входного сигнала, определение алгоритмической устойчивости схемы (например эффект состязания в схеме) и прочее...

37. Системы логического моделирования.

Системы логического моделирования: в зависимости от способа хранения информации на схеме различают

1)компилятивные (2 способа)

1.описание модели выполняется на одном из специализировных языков(например VHDL, этот способ удобнее)

2.использование языков высокого уровня, при этом модель строится с учетом синтаксиса и семантики языка. (не требуют знаний специализированных языков)

2)интерпретативные

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

Языки систем моделирования ориентированы на описания :

- событий - процессов - активностей

Основой любой системы моделир. является внутренняя система (планировщик), кот. организует последов. действий.

Ф-ции планировщ. : 1. Отслеживание времени работы

2. Отслеживание последовательности действий

В любой системе в любую единицу врем. может вып. одно или несколько действий. Необходимо обесп. моделир. паралел. во времени действий.

Во время T=t1 во времени у нас выполняется три активности. Мы будем в это время в реал. времени вып. эти активности.

Должно выполнятся условие- свободно ли устройство, на котором вып. данная активность.

Языки ориентированные на описание активностей.

Планировщик, или сист. моделирования , содержит списки активностей и время (CSL).

В яз. ориент. на опис. событий. планировщик содержит переменную время и списки (очереди) событий. все события заданы явно в виде процедур.

Языки ориент. на опись процессов. События следования задаются неявно, а события изменения зад. явно. Здесь задаются активности. Процесс – совокупность активностей.

В зависимости от ориент. языка планировщ. содержат списки активностей, событий, процессов и еще очереди.

38. Модели элементов в системе логического моделирования.

E = (φ,A,Δ). A-алфавит в котором работает схема. Δ-динамические параметры с задержками, φ-функция.

Логический элемент можно пердставить как 2 блока: логический и динамический.

По степеням адекватности модели бывают :

1) Л-модели.учитывают только логику , но не задержку =>позволяют определить достежимость определенного состояния.

2)ЛД-модели.(логика и динамика)

3) ЛИД - модели (учитывается также инерционная задержка)

Инерционная задержка: для обеспечения срабатываний необходимо чтобы длительность входных сигналов была не меньше τd (время срабатывания)

—

Самая простая модель, двоичная, когда алфавит содержит лишь булевые значения A = {0,1}

Для расширения возможностей данной модели, добавляют ещё один сигнал, это уже троичная модель сигналов A = {0,1,X}

A4={0,1,λ,ε}, где λ - переключение из 0 в 1, а ε - переключение из 1 в 0.

A5={0,1,λ,ε ,X}, A6={0,1,λ,ε,p,h}, где p - статистический сбой, h - динамический сбой.

A7={0,1,λ,ε,p,h,X}, и так далее. Чем больше модель, тем адекватнее анализ. Однако тем более громоздко идёт вычисление.

39. Модели сигналов в системе логического моделирования.

Различают статические и динамические модели.

В статических моделях во всех схемах предполагается, что все временные параметры идеальны. В статическом моделировании есть входные параметры, получаем таблицу выходных параметров и можно проверить правильность выполнения функции путем сравнения с ожидаемым.

В динамических моделях необходимо учитывать в схеме задержку времени переключения на каждом элементе (Δt – разное), поэтому появляются гонки. В результате появляются неправильные выходные сигналы. Различают статические состязания и динамические.

Статические состязания: $Y(t_1) = f(x_1)$, $Y(t_2) = f(x_2)$. $Y(t_1) = Y(t_2)$, но в момент времени $t_1 < \tau < t_2$ появляется $Y(t_1) \neq Y(t_2)$. В зависимости от того, приведут ли состязания к алгоритмическому переходу различают опасные и неопасные состязания. Опасные состязания приводят к неправильному алгоритмическому переходу.

В динамических состязаниях $Y(t_1) \neq Y(t_2)$, но в момент времени $t_1 < \tau_1 < \tau_2 < t_2$ появляется $Y(t_1) \neq Y(\tau_1)$, $Y(\tau_1) \neq Y(\tau_2)$, $Y(\tau_2) \neq Y(t_2)$. Различают функциональные, логические состязания и состязания на уровне логических элементов. Функциональные – если на переходе Y_1 в Y_2 возникает два алгоритмических перехода. Логические – могут дать на выходе неалгоритмический переход. Состязания возникают в логических элементах, выполняющих сложные функции, не за счет гонок, а за счет состязаний внутри элемента

Самая простая модель, двоичная, когда алфавит содержит лишь булевые значения A = {0,1}

Для расширения возможностей данной модели, добавляют ещё один сигнал, это уже троичная модель сигналов A = {0,1,X}

A4={0,1,λ,ε}, где λ - переключение из 0 в 1, а ε - переключение из 1 в 0.

A5={0,1,λ,ε ,X}, A6={0,1,λ,ε,p,h}, где p - статистический сбой, h - динамический сбой.

A7={0,1,λ,ε,p,h,X}, и так далее. Чем больше модель, тем адекватнее анализ. Однако тем более громоздко идёт вычисление.

Различают статические состязания (предполагают появлене одного единичного (нулевого) сигнала на выходах 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 появляются сигналы у(тау)!=у(т1), то такие состязания статическими состязаниями т1< тау <т2.

Динамическое состязание – несанкционирование изменение сигналов на выходе несколько раз.

Функционирующее состязание возникает,если при переходе на входе от х1 к х2 возникает больше 2-ух алгоритмических переходов.

40. Модели схем в системе логического моделирования.

Модель схемы описывается формулой S = { E , ksi}, где E - множество моделей элементов, а ksi - множество связей между этими элементами. Схемы бывают двух типов, асинхронные и синхронные. Асинхронные модели учитывают задержки на элементах и их переходные процессы, по-этому можно легко проанализировать и адекватно работоспособность схемы в любой момент времени. Однако этот процесс трудоёмкий, и чаще всего не так уж и необходим. Синхронные модели схем считается что в таких схемах задержек нету, всё работает синхронно, поэтому время выполнения анализа таких схем быстрее и проще.

Двузначные модели наиболее экономичны, но с их помощью можно решать лишь ограниченный круг задач, например, выявление грубых ошибок в построении функциональной схемы. А для определения с помощью синхронных моделей рисков сбоя вместо двухзначных моделей необходимо использовать трех- и более значные модели.

41. Риски сбоя в схемах и методы их обнаружения.

Риск сбоя – это возможность появления истинных сигналов. Различают статический и динамический риск сбоя.

Статический риск сбоя – это когда возможно однократное изменение сигнала на выходе элемента при правильном его функционировании (то есть при физически исправном устройстве). для определения с помощью синхронных моделей рисков сбоя вместо двухзначных моделей необходимо использовать трех- и более значные модели.

Отразить статический риск сбоя можно с помощью троичных моделей. В этих моделях переменные могут принимать значения из множества $\{0,1,x\}$ где x – неопределенное состояние. Это неопределенное состояние возникает во время переходных процессов как промежуточное при переключениях из состояния 1 в состояние 0 (или наоборот).

Динамический риск сбоя представляет собой опасность многократного изменения выходящей переменной вместо правильного однократного изменения.

выявить динамичный режим сбоя позволяют пятизначные модели. То есть пятизначные величины принимают значения из множества {0,1,λ,ε ,X}, где λ – переход из 1 в 0, ε – из 0 в 1.

42. Общая характеристика алгоритмов моделирования.

Алгоритмы моделирования бывают следующих типов:

1. Итерационные.
    1.1 Простая итерация
    1.2 Ускоренная итерация (Зейделя)
2. Событийные.

Каждый из типов алгоритмов могут в свою очередь быть паралельными или последовательными.

Суть 1 в том, что на каждом шаге вычисляется значение каждого элемента который есть в схеме.

Суть 2 в том, что на каждом шаге просчитываются только те элементы, на которых могут произойти изменения.

Многозначные модели сигналов чаще используются для итерационных алгоритмов моделирования. Параметры элементов - нединамические. Быстрота - количество итераций, необходимых для расчёта схемы.

В событийных - параметры элементов динамические, но редко используются многозначные модели сигналов.

43. Алгоритм простой итерации (2-ая модель сигналов)

Подразумевается, что вторая модель сигналов, это модель при котором существует лишь два уровня сигнала: A2 = {0,1}

$$Y_{i,j}^{k} = F(X^{k}, Y_{i,j-1}^{k}); k = \overline{(1,m)}; i = \overline{(1,n)}$$

, где Y - значения выходных сигналов.

X - входные наборы

i - номер элемента схемы

j - номер итерации на к-том наборе

АЛГОРИТМ

1. Задать начальное состояние схемы.
2. Подать входной набор
3. Просчитать значение выхода каждого элемента схемы.
4. Перейти к 1 пункту и в качестве исходных данных для каждого элемента использовать значение входного набора и значение выходных сигналов (как начальное состояние), полученной на предыдущей итерации.
5. Повторять итерации до тех пор, пока схема не перейдёт в устойчивое состояние.

(Устойчивое состояние определяется одинаковыми значениями двух последовательных итераций)

6. Если схема не перешла в устойчивое состояние быстрее чем максимальное количество шагов, схема неустойчивая (максимальное количество шагов, это критический путь, т.е самая длинная цепочка в схеме + 1).
7. Затем подать следующий входной набор.
8. Промоделировать таким образом все данные входные наборы.

44. Алгоритм ускоренной итерации (2-ая модель сигналов)

Подразумевается, что вторая модель сигналов, это модель при котором существует лишь два уровня сигнала: A2 = {0,1}

$$Y_i^k = F(X^k, Y_{i,j}^k, ..., Y_{i-1,j}^k, Y_{i,j}^{k-1}, ..., Y_{i-1,j}^{k-1}); k = \overline{(1,m)}; i = \overline{(1,n)}$$

, где Y - значения выходных сигналов.

X - входные наборы

i - номер элемента схемы

j - номер итерации на к-том наборе

Если нумерация элементов в схеме неправильная нужно перенумеровать элементы. На каждой итерации основное берём из предыдущей и также используем всё что уже известно на данный момент.

АЛГОРИТМ

1. Задать начальное состояние схемы.

2. Подать входной набор

3. Просчитать значение выхода каждого элемента схемы с учётом текущего значения уже посчитанных элементов схемы (а не с учётом предыдущих сигналов того элемента как это было в простой итерации).

4. Перейти к 1 пункту и в качестве исходных данных для каждого элемента использовать значение входного набора и значение выходных сигналов (как начальное состояние), полученной на предыдущей итерации.

5. Повторять итерации до тех пор, пока схема не перейдёт в устойчивое состояние.

(Устойчивое состояние определяется одинаковыми значениями двух последовательных итераций)

6. Если схема не перешла в устойчивое состояние быстрее чем максимальное количество шагов, схема неустойчивая (максимальное количество шагов, это количество обратных связей + 2).

7. Затем подать следующий входной набор.

8. Промоделировать таким образом все данные входные наборы.

45. Алгоритм простой итерации (3-ая модель сигналов)

Подразумевается, что вторая модель сигналов, это модель при котором существует лишь два уровня сигнала: A3 = {0,1,X} (X - промежуточный сигнал между 0 и 1)

$$Y_{i,j}^k = F(X^k, Y_{i,j-1}^k); k = \overline{(1,m)}; i = \overline{(1,n)}$$

, где Y - значения выходных сигналов.

X - входные наборы

i - номер элемента схемы

j - номер итерации на к-том наборе

В данном алгоритме между каждыми соседними входными наборами выставляется промежуточный набор, в котором соответствующие сигналы если меняются между собой, то в этом новом наборе будут иметь значение X.

АЛГОРИТМ

1.2. Задать начальное состояние схемы и подать входной набор

3. Просчитать значение выхода каждого элемента схемы.

4. Перейти к 1 пункту и в качестве исходных данных для каждого элемента использовать значение входного набора и значение выходных сигналов (как начальное состояние), полученной на предыдущей итерации.

5. Повторять итерации до тех пор, пока схема не перейдёт в устойчивое состояние.

(Устойчивое состояние определяется одинаковыми значениями двух последовательных итераций)

6. Если схема не перешла в устойчивое состояние быстрее чем максимальное количество шагов, схема неустойчивая (максимальное количество шагов, это критический путь, т.е самая длинная цепочка в схеме + 1).

7. Затем подать следующий входной набор.

8. Промоделировать таким образом все данные входные наборы.

46. Алгоритм ускоренной итерации (3-ая модель сигналов)

Подразумевается, что вторая модель сигналов, это модель при котором существует лишь два уровня сигнала: А3 = {0,1,X} (X - промежуточный сигнал между 0 и 1)

$$Y_i^k = F(X^k, Y_{i,j}^k, ..., Y_{i-1,j}^k, Y_{i,j}^{k-1}, ..., Y_{i-1,j}^{k-1}); k = \overline{(1,m)}; i = \overline{(1,n)}$$

, где Y - значения выходных сигналов.

X - входные наборы

i - номер элемента схемы

j - номер итерации на к-том наборе

Если нумерация элементов в схеме неправильная нужно перенумеровать элементы. На каждой итерации основное берём из предыдущей и также используем всё что уже известно на данный момент. В данном алгоритме между каждыми соседними входными наборами выставляется промежуточный набор, в котором соответствующие сигналы если меняются между собой, то в этом новом наборе будут иметь значение X.

АЛГОРИТМ

1, 2. Задать начальное состояние схемы. Подать входной набор

3. Просчитать значение выхода каждого элемента схемы с учётом текущего значения уже посчитанных элементов схемы (а не с учётом предыдущих сигналов того элемента как это было в простой итерации).

4. Перейти к 1 пункту и в качестве исходных данных для каждого элемента использовать значение входного набора и значение выходных сигналов (как начальное состояние), полученной на предыдущей итерации.

5. Повторять итерации до тех пор, пока схема не перейдёт в устойчивое состояние.

(Устойчивое состояние определяется одинаковыми значениями двух последовательных итераций)

6. Если схема не перешла в устойчивое состояние быстрее чем максимальное количество шагов, схема неустойчивая (максимальное количество шагов, это количество обратных связей + 2).

7. Затем подать следующий входной набор.

8. Промоделировать таким образом все данные входные наборы.

47. Алгоритм ранжирования

Нулевой ранг выставляется всем входным контактам. Далее в следующий ранг записываются все элементы, которые имеют связь только с входными контактами. Потом те которые могут иметь связь с ранжированными элементами и входными контактами.

АЛГОРИТМ:

1. Записываем нулевой ранг (входные контакты)

2. Просматриваем всю схему, ищем все элементы i-того ранга.

3. Повторяем, пока не отранжируем все.

ДАННЫЕ АЛГОРИТМ НЕ ПРИМЕНЯЕТСЯ ПРИ СХЕМАХ С ОБРАТНЫМИ СВЯЗЯМИ.

Для схем с обратными связями используется алгоритм условного ранжирования. В нём алгоритм тот же, до тех пор пока не встречаем обратную связь, тогда мы разрываем эту связь.

48. Событийный алгоритм моделирования (статическая модель элементов)

В этом методе событием называют изменение любой переменной модели. В сложных логических схемах на каждом такте синхронизации обычно происходит переключение всего лишь 2-3% логических элементов и, соответственно, в событийном методе в несколько раз уменьшаются вычислительные затраты по сравнению с пошаговым моделированием. Событийное моделирование основано на следующем правиле: обращение к модели логического элемента происходит лишь в том случае, когда на входах этого элемента произошло событие.
Для достижения этого правила используют списки (таблицы) текущих событий (ТТС) и будущих событий (ТБС).

АЛГОРИТМ:

1. Устанавливаем исходное состояние схемы. Если все нули, то происходит моделирования схемы на любом наборе по любому итерационному методу.
2. В ТБС заносятся номера тех элементов, входы которых поменяли значения.
3. В ТТС заносится ТБС, ТБС обнуляется, просчитывается значения тех элементов которые находятся в ТТС. Возвращаемся в пункт 2.
4. Процесс повторяется, пока обе таблицы не будут пусты.
5. Как только таблицы пусты, подать следующий входной набор.

Таким образом моделируется статическая синхронная модель элементов.
(также называется Л-моделью)

49. Событийный алгоритм моделирования (ЛД-модель элемента)

В этом методе событием называют изменение любой переменной модели. В сложных логических схемах на каждом такте синхронизации обычно происходит переключение всего лишь 2-3% логических элементов и, соответственно, в событийном методе в несколько раз уменьшаются вычислительные затраты по сравнению с пошаговым моделированием. Событийное моделирование основано на следующем правиле: обращение к модели логического элемента происходит лишь в том случае, когда на входах этого элемента произошло событие.
Для достижения этого правила используют списки (таблицы) текущих событий (ТТС) и будущих событий (ТБС). Для моделирования ЛД-модели элемента используют также временной параметр задержки переключения элемента. Добавляется к таблицам также столбик с значением Т (текущее время моделирования).

АЛГОРИТМ:

1. Устанавливаем исходное состояние схемы. Если все нули, то происходит моделирования схемы на любом наборе по любому итерационному методу.
2. Выбираются те элементы, входы которых изменились (хотябы один вход), а из этого списка выбираются те элементы которые переключатся в ближайшее время (к текущему).
3. В ТТС заносится ТБС, ТБС обнуляется, просчитывается значения тех элементов которые находятся в ТТС. Возвращаемся в пункт 2.
4. Процесс повторяется, пока обе таблицы не будут пусты.
5. Как только таблицы пусты, подать следующий входной набор.

50. Паралельная реализация итерационных алгоритмов

Паралельные алгоритмы - это последовательная реализация и просчёт сразу нескольких наборов в один и тот же момент времени на одной схеме. (написать пример и чтото про другие итерационные алгоритмы)

51. Последовательная реализация событийных алгоритмов

Последовательная реализация является приближённой и не содержит таблиц (ТТС и ТБС) и соответственно время моделирования сокращается. В данной реализации просчитываются элементы схемы последовательно от начала до конца.

1. Устанавливаем исходное состояние схемы. Если все нули, то происходит моделирования схемы на любом наборе по любому итерационному методу.
2. Выбираются те элементы, входы которых изменились (хотябы один вход), а из этого списка выбираются те элементы которые переключатся в ближайшее время (к текущему).
3. Эти элементы расчитываются и переходят к пункту 2.

52. 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. (проверяет исправен объект или нет.) Под *диагностикой* подразумевается процедура локализации неисправности объекта (определяет место неисправности). Все процедуры сводятся к одной какой-то схеме:

Нельзя гарантировать, что эталон не будет сбоить.

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. Всего существует три типа контроля: *тестовый контроль* (ТК), *функциональный контроль* (ФК) и *параметрический контроль* (ПК).

- При тестовом контроле необходимо, как минимум дополнительная память для хранения самих тестов.
- При функциональном – аппаратура, которая выполняет функции.
- При параметрическом – аппаратура сбора информации о значениях параметров контроля. При функциональном аппаратура встроенная, при параметрическом – внешняя.

53. Избыточность и трудоёмкость 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.

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.

54. Общая характеристика функционального контроля СВТ

Функциональный контроль определяет правильность выполнения функций.

Необходимо определить правильность выполнения функций – функция передачи информации, арифметические и логические функции.

Контроль передачи информации выполняется при помощи специально подобранных кодов. Часто используют различные коды, которые предполагают передачу информации, состоящую из двух частей, информационная и кодовая проверка. Такой контроль мало эффективен.

Чаще всего используется система счисления в остатках, которая по заданным модулям определяет остатки от деления этого числа.

Для исправления ошибок используются специальные коды:

Хемминга: Они проводят остатки по mod 2 некоторого количества разрядов. Соответственно количество дополнительных разрядов больше. Если неправильно передали информацию, то при составлении кода будет 3 подряд несоответствия и в общем из них разряде в информации нужно изменить.

для арифметических операций:

A+B=C: (A)mod + (B)mod = (C)mod

A*B=C: (A)mod * (B)mod = (C)mod

A/B = Y + W: (A)mod = ((Y)mod + (W)mod)mod * (B)mod

для логических операций: сдвиги.

Обычно либо повторным, либо параллельным сдвигом.

55. Общая характеристика параметрического контроля СВТ

Системы параметрического контроля строятся на основе оценки состояния ОК по некоторым косвенным признакам и обладают преимуществом, выражающимся в не повреждающем характере испытательных воздействий.

Обычно тесты предназначаются для выявления одиночных неисправностей (под одиночной понимается неисправность с точностью до одного элемента, на выходе которого она может быть зафиксирована). Диагностика множественных неисправностей это другой, более сложный, класс задач.

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.

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» – неправильном функционировании, вызванном определенным сочетанием значений входных переменных.

Однако параметрический контроль (диагностика) не требует детального структурного описания ОК.

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

56. Общая характеристика тестового контроля СВТ

Тестовый контроль – специально подобранная тестовая задача, входные воздействия которой позволяет определить исправность объекта. Выполняется в специально отведенные промежутки времени при помощи специально подобранных воздействий. Тестовый контроль проводится при нефункционировании объекта. Все тесты строятся на определении логических неисправностей.

Обычно тесты строятся для выявления одиночных неисправностей (на одном элементе). Для выявления кратных неисправностей необходимо использовать специальные методы.

Элементарная проверка состоит из двух векторов T = {X, Y} X- вектор входных воздействий, Y – вектор реакции. Тестовая последовательность содержит множество проверок, позволяющих обнаружить все неисправности в заданном классе.

В интегральной схеме можно выделить ряд неисправностей, которые отличны от константных неисправностей: *ближайшее соседство* (паразитная связь между ближайшими элементами цифровых устройств изза топологии размещения элементов), *соседство* (топологически не рядом размещенные элементы, паразитическая связь между ними), *паразитная задержка, увеличение времени прохождения сигнала через объект, чувствительность к двоичным наборам* (неправильное функционирование устройства, вызванное сочетанием двоичных значений набора), *насыщение шины*.

57. Контроль передачи информации

Для контроля процесса преобразования информации, в основном, используются равномерные избыточные коды. Избыточность равномерного разделимого кода определяется длиной подслова, соответствующего контрольному коду. Эту избыточность стремятся по возможности минимизировать в силу следующих причин:

- При хранении в памяти всего избыточного кода требуется большое количество физических элементов памяти. Тоесть аппаратурные затраты будут кратно возрастать.
- При хранении в памяти информации 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.
- Вероятность возникновения кратной ошибки при выполнении операций обработки информации принимается небольшой (на текущий момент), поэтому применяют коды, обнаруживающие только ограниченное множество ошибок.

Контроль передачи информации выполняется при помощи специально подобранных кодов. Часто используют различные коды, которые предполагают передачу информации, состоящую из двух частей, информационная и кодовая проверка. Такой контроль мало эффективен.

Чаще всего используется система счисления в остатках, которая по заданным модулям определяет остатки от деления этого числа.

Для исправления ошибок используются специальные коды Хемминга: Они проводят остатки по mod 2 некоторого количества разрядов. Соответственно количество дополнительных разрядов больше. Если неправильно передали информацию, то при составлении кода будет 3 подряд несоответствия и в общем из них разряде в информации нужно изменить.

58. 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 вытекает из теорем:

(A)mod q+ (B)mod q = (C)mod q, (A)mod q* (B)mod q = (C)mod q.

Поэтому: A+B=C: (A)mod q+ (B)mod q = (C)mod q

A*B=C: (A)mod q* (B)mod q = (C)mod q

A/B = Y + W: (A)mod q = ((Y)mod q+ (W)mod q)mod q * (B)mod q

Значением q определяется полнота контроля. При увеличении значения модуля увеличивается число кратных ошибок, обнаруживаемых системой контроля, однако, при этом возрастает и сложность кодирующей и контролирующей аппаратуры. Избыточность будет минимальной при условии $q = r \pm 1$, где r - основание системы счисления, т.е. для СВТ значение q принимается равным «3».

59. Требования предъявляемые к тестам.

1) Достаточная полнота контроля. Полный тест - тест, реализация которого позволяет обнаружить все возможные неисправности из заданного класса неисправностей. Колличественная оценка полноты контроля есть отношение колличества выявляемых неисправностей к общему числу возможных неисправностей заданного класса.

2) Приемлемое время испытаний объекта контроля. Время контроля непосредственно связанно к количеством элементныхх проверок или с емкосной сложностью теста прямой 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.

3) Малая трудоемкость генерации тестов. Создание тестов, эффективных по показателю полноты и количеству элемементных проверок - сложная задача.

4) Глубина диагностирования. Очевидным требованием к диагностическим тестам является локализация места неисправности с точностью до конкретного сменного элемента или до определённой области применимости с ограничением функций для “неразборных” объектов, например БИС.

60. Классификация методов тестового контроля.

В зависимости от того как генерируется последовательность входных векторов, различают:

- исчерпывающие тесты – полный перебор входных воздействий;
- детерминированные тесты – задаются полные, близкие к минимальным тесты;
- вероятностные – случайные последовательности генерируются при помощи генераторов случайных и псевдослучайных чисел.

Используют методы сжатия результатов. Эти методы называют методами компактного тестирования.

Тестовый – специально подобранная тестовая задача, входные воздействия которой позволяет определить исправность объекта. Выполняется в специально отведенные промежутки времени при помощи специально подобранных воздействий.

Тестовый контроль проводится при не функционировании объекта. Все тесты строятся на определении логических неисправностей.

61. Методы сжатия реакций ОК

Принятие решения о техническом состоянии ОК основывается на анализе выходной последовательности в матричном представлении $Y=\left[y_{t,i}\right]$. 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 $W=\langle w_1,w_2,...,w_n\rangle$ код $K=\langle k_1,k_2,...,k_m\rangle$ ($m<n$). 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 $m<n$, то, естественно, возникает вопрос о достоверности компактного тестирования, т.е. о вероятности того, что примененный метод сжатия позволяет обнаружить все предполагаемые неисправности ОК. При этом анализируется влияние, которое оказывает неисправность на вид выходной последовательности и то, каким образом это влияние отображается в код K.

62. Функции счёта

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

для *m=0*:

а) функция счета единичных значений результатов

$$S_0^1(R)=\sum_{i=1}^{n} r_i;$$

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

$$S_1^2(R)=\sum_{i=2}^{n}\left(r_{i-1}\oplus r_i\right);$$

для *m=1*:

в) функция счета числа повторений значений результатов

$$S_1^3(R)=\sum_{i=2}^{n}\overline{\left(r_{i-1}\oplus r_i\right)};$$

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

$$S_1^4(R)=\sum_{i=2}^{n}\left(\bar{r}_{i-1}r_i\right);$$

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

$$S_1^5(R)=\sum_{i=2}^{n}\left(r_{i-1}\bar{r}_i\right).$$

63. Контрольные суммы

Контрольные суммы- это один из основных методов сжатия выходной информации
Пусть задано упорядоченное множество из *n* *m*-разрядных чисел $\{u_i\}$, где $i=1,2,...,n$; $u_i=u_{i1},u_{i2},...,u_{im}$ соответствующее выходной последовательности ДУ. Используются следующие способы суммирования:
а) поразрядное суммирование по модулю 2:

$$K_2(u_1,u_1,...,u_n)=v_1,v_2,...,v_m;\quad v_i=\bigoplus_{i=1}^{n}u_{j_i};$$

б) арифметическое суммирование по различным модулям:

$$K_M(u_1,u_1,...,u_n)=(u_1+u_2+...+u_n)_{\mathrm{mod}\ M},$$

причем
$M=n(2^m-1)$ - полная арифметическая сумма;
$M=2^m$ - арифметическая сумма без учета переноса из старшего разряда;
$M=2^m-1$ - арифметическая сумма с циклическим переносом в младший разряд.

64. Синдром

Синдром- это один из основных методов сжатия выходной информации.
Синдромное тестирование используется при исчерпывающем компактном тестировании.
Синдромом булевой функции называется число

$$S=K/2^n,$$

где *K* – 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.
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).
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.
Для реализации синдромного тестирования комбинационные схемы должны проектироваться таким образом, чтобы синдром исправной схемы отличался от неисправной.

65. Спектральные коэффициенты

Сжатие информации с помощью спектральных коэффициентов используется при исчерпывающем компактном тестировании комбинационных схем.
Пусть на вход схемы поступает набор $x_1,x_2,...,x_n$, $i_1,i_2,...,i_t$ – номера тех разрядов входного набора функции $F(x_1,x_2,...,x_n)$, зависимость *F* от которых необходимо определить [50]. *Функциями Уолша* w_i называются функции, принимающие значения ±1 и вычисляемые в соответствии с формулой:

$$w_{i_1,i_2,...,i_l}(x)=\prod_{j=1}^{l}R_{i_j}(x),$$

где $R_{i_j}(x)$ - *функция Радемахера:* $R_{i_j}(x)=(-1)^{x_{i_j}}$.

Всего имеется 2^n функций Уолша. Например, для функции $F(x_1,x_2,x_3)$ имеется восемь функций Уолша: w_0, w_1, w_2, w_{12}, w_3, w_{13}, w_{23}, w_{123}.
Спектральным коэффициентом или коэффициентом Уолша называется функция

$$S(i_1,...,i_l)=\sum_{x=0}^{2^m-1}F(x)\,w_{i_1,i_2,...,i_l}(x),$$

показывающая меру зависимости значения функции от суммы по модулю 2 разрядов $x_{i_1},...,x_{i_l}$ *входного набора;* $S(i_j)$ *– зависимость от* i_j*-го разряда.*

Коэффициенты Уолша можно вычислить также по следующей формуле [50]: $S=T_n\cdot F$

где T_n есть $2^n\times2^n$ – трансформационная матрица, определяемая следующим образом:

$$T_n=\begin{vmatrix}T_{n-1} & T_{n-1}\\ T_{n-1} & -T_{n-1}\end{vmatrix},$$

а T_0=[l]. Строки матрицы T_n представляют собой значения функций Уолша.

66. Вероятностные тестирования

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

Одна из возможных схем вероятностного некомпактного тестирования приведена на рис.3.8. 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 (рис.3.9).

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.

67. Общая характеристика методов синтеза детерминированных тестов.

Задача синтеза тестов относится к классу универсальных переборных задач. Анализируя весь спектр материала, представленного в предыдущей главе, можно прийти к выводу о том, что задача синтеза детерминированных тестов формально может быть поставлена с использованием наиболее общих моделей ОД и моделей неисправностей, т.е. с использованием понятий статуса системы и пространства состояний.

Процесс вычисления тестов в общем случае состоит из следующих этапов:

- определение списка неисправностей;
- вычисление тестовых наборов (элементарных проверок) для неисправностей из этого списка;
- минимизация теста.

Поиск эквивалентных неисправностей можно не производить в том случае, когда мощность предполагаемого класса неисправностей невелика. Однако, такие ситуации достаточно редки, а решение задачи поиска эквивалентных неисправностей позволяет в значительной степени сократить сложность как синтеза, так и реализации теста.

Методы синтеза детерминированных тестов можно разделить на две группы.

- методы *моделирования неисправностей* (основаны на анализе входного алфавита ОД)
- методы *вычисления теста* (исходными данными является неисправность $\Phi_j \in C$, а результатом – тестовый набор (множество тестовых наборов) $P_X(\Phi_j \setminus F)$)

68. Понятие "тест", "контролирующий тест", "диагностирующий тест". Методы минимизации тестов.

Тест – элементарная проверка, которая состоит из 2-х векторов (вектор входных реакций и выходных реакций). Это конечное множество элементарных проверок

контролирующий тест - тест, который служит для проверки исправности или работоспособности проверяемого блока.

диагностический тест - тест, предназначенный для обнаружения места неисправности.

69. Метод таблиц истинности синтеза тестов

Метод таблиц истинности использует моделирование ОД с неисправностями. Анализируются все из 2^n входных наборов схемы, где n — число входов ОД.

После определения классов эквивалентных неисправностей выполняется моделирование, результатом которого является таблица функций различения (ТФР), задающая отношение τ. Затем находится минимальное покрытие этой таблицы.

{влепить какойто пример}

70. Метод активизации путей синтеза тестов

Основой многих из методов вычисления тестов является идея активизации пути.

Под активизацией пути понимается выбор последовательности линий схемы от места предполагаемой неисправности до внешних выходов. Значения сигналов на линиях, входящих в активизированный путь, для исправной схемы и при наличии неисправности должны быть различными. Для того, чтобы обеспечить это условие, необходимо зафиксировать определенное состояние линий, которые не входят в активизированный путь. Другими словами, необходимо обеспечить режим «прозрачности» схемы вдоль активизированного пути.

Несмотря на все отличия методов первой и второй группы, в результате их применения можно получить отношение $\tau \subseteq T \times C$, в котором T соответствует множеству синтезированных тестовых наборов, а C – предполагаемому классу неисправностей. Отношение τ задает разбиение множества C на подмножества, обнаруживаемых каждым из тестовых наборов неисправностей и может быть представлена бинарной таблицей, строки которой соответствуют элементам множества T, а столбцы – элементам из C. В этом случае задача минимизации теста эквивалентна задаче поиска минимального покрытия, методы решения которой хорошо исследованы..

71. D-алгоритм синтеза тестов

D-алгоритм является одним из первых разработанных методов синтеза детерминированных тестов для комбинационных схем, в основу которого положена идея активизации пути. Вычисление тестового набора основано на создании условий проявления неисправности и активизации пути от места ее проявления до выхода схемы. Для определения условий проявления неисправностей достаточно анализировать описание элементов схемы.

Необходимо синтезировать множество тестовых наборов, обнаруживающих ее, т.е. для каждого из элементов схемы необходимо найти тест. Используемый в D-алгоритме метод активизации пути предполагает наличие двух стадий. На первой из них определяются условия активизации элементов, на второй осуществляется выбор таких условий для каждого из элементов, которые были бы непротиворечивы для всей схемы в целом.

Условие активизации элемента E задается парой его состояний $\langle e_i, e_j \rangle$, для которой выполняется условие: $\langle e_i, e_j \rangle$: $\left(P_X(e_i) \neq P_X(e_j)\right) \Rightarrow \left(P_Y(e_i) \neq P_Y(e_j)\right)$. Таким образом, определить условия активизации элемента E означает найти множество $\Delta \subseteq (X_e \times Y_e)^2$

Отношения τ и Δ определены в пространстве $(X_e \times Y_e)^2$, следовательно для их описания может использоваться один математический аппарат.

В D-алгоритме для описания элементов множеств τ и Δ используется отображение, введенное Ротом и названое им d-операцией.

Таким образом, описание элемента состоит из:

- множества кубов неисправности – множества τ, заданного в виде (5.8);
- множества кубов распространения (D-кубов) – множества Δ, элементы которого представляют собой описание элемента в алфавите $\{0,1,D,\overline{D}\}$;
- множества кубов вырожденного покрытия – функционального описания элемента в алфавите $\{0,1,x\}$.

Если такое описание известно для всех элементов схемы, то действия, выполняемые на второй стадии активизации пути сводятся к решению переборной задачи на множествах кубов неисправности, кубов продвижения и кубов вырожденного покрытия. При этом, определяя условия непротиворечивости выбранных кубов, исходят из того, что состояние каждой линии схемы должно однозначно определять состояние множества входов-выходов, ассоциированных с ней. Алфавит состояний для линии определяется множеством $\{0,1,D,\overline{D},x\}$.

72. 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,..., $\overline{x_i}$,..., 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 (xi = 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 ($\overline{z}$× 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.

73. Цепной метод поиска булевой производной

$$\frac{dy}{dx_i} = y(x_1,...,x_{i-1},0,x_{i+1},...,x_n) \oplus y(x_1,...,x_{i-1},1,x_{i+1},...,x_n)$$

Выполнив суперпозицию можна записать

$$Y = y(x_1,...,x_{k-1} * y_1(x_k,...,x_i,...,x_n))$$

$$\frac{dy}{dx_i} = \frac{dy}{dy_1} \cdot \frac{dy_1}{dx_i}$$

$$\frac{dy}{dx_i} = \frac{dy}{dy_1} \cdot \frac{dy_1}{dy_2} \cdot \ldots\ldots \cdot \frac{dy_m}{dx_i}$$

Идея метода – при разумном выборе функции $y_1 \ldots y_m$ все производные в правой части берутся достаточно просто, например по правилам 7 и 8. Но это справедливо в том случае,

если при каждой суперпозиции находится единственная ф-ция $y_1 \ldots y_m$ которая зависит от x_i. Это соответствует комбинационной схеме без разветвлений.

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

{ВЫПИСАТЬ 73 ВОПРОС КОТОРЫЙ ВЫШЕ}

Вычисления можна упростить используя префиксную форму

$$y = (((x1 + x2)_1 x3x4)_3 (x5 + x6)_2)_4$$

$$y = (_4 \wedge (_3 \wedge (_1 \vee (x1x2)_1 x3x4)_3 (_2 \vee x5x6)_2)_4$$

Аргументы в скобках называются списком аргументов

Правило интерпретируется

$$\frac{d(\overline{\wedge} список\_аргументов)}{d(аргумент\_из\_списка)} = \wedge список\_аргументов\_без\_аргументаив\_списка$$

$$\frac{d(\overline{\vee} список\_аргументов)}{d(аргумент\_из\_списка)} = \overline{\wedge} список\_аргументов\_без\_аргументаив\_списка$$

75. Метод эквивалентных нормальных форм синтеза тестов

Этот метод основан на представлении булевой функции и виде эквивалентной нормальной формы (ЭНФ), описывающей конкретную реализацию схемы. Поскольку ЭНФ представляет собой сумму логических произведений, она соответствует гипотетической схеме нескольких И— ИЛИ. Каждой схеме И соответствует один терм ЭНФ. Из такого представления ЭНФ становится очевидным, что для выявления неисправностей, связанных с переменной x_i, входящей в какой-либо терм ЭНФ, необходимо выполнение следующих условий:

1. равенство нулю всех термов, кроме содержащего переменную x_i;
2. равенство единице всех переменных терма, в который входит тестируемая переменная x_i.

Выполнение этих условий обеспечивает тождественное равенство $f(X) \equiv x_i$ и, как следствие этого, выявление неисправностей, связанных с этой переменной, так как неисправность переменной приведет к изменению сигнала на выходе схемы.

Эквивалентная нормальная форма, как и обычная нормальная, вычисляется методом подстановки, с той лишь разницей, что избыточные термы не исключаются, так как они характеризуют конкретную реализацию схемы.

При построении тестов важно не только обеспечить проверку входных переменных, но и всех путей, т.е. необходимо обеспечить проверку одной и той же переменной в разных термах, которым соответствуют разные пути в схеме.

76. Синтез тестов для последовательных 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.

77. Событийный алгоритм моделирования (ЛИД - модель элемента)

В этом методе событием называют изменение любой переменной модели. В сложных логических схемах на каждом такте синхронизации обычно происходит переключение всего лишь 2-3% логических элементов и, соответственно, в событийном методе в несколько раз уменьшаются вычислительные затраты по сравнению с пошаговым моделированием. Событийное моделирование основано на следующем правиле: обращение к модели логического элемента происходит лишь в том случае, когда на входах этого элемента произошло событие.
Для достижения этого правила используют списки (таблицы) текущих событий (ТТС) и будущих событий (ТБС).
ЛИД - модель - полная модель задержки схемы. Выявляет временную и амплитудную фильтрацию. Разброс входных параметров или коротких тактовых импульсов могут привести к временной или амплитудной фильтрации. ЛИД модель определяет где и когда должна произойти фильтрация.
АЛГОРИТМ:

1. Устанавливаем исходное состояние схемы. Если все нули, то происходит моделирования схемы на любом наборе по любому итерационному методу.
2. Выбираются те элементы, входы которых изменились (хотябы один вход), а из этого списка выбираются те элементы которые переключатся в ближайшее время (к текущему).
3. В ТТС заносится ТБС, ТБС обнуляется, просчитывается значения тех элементов которые находятся в ТТС. Возвращаемся в пункт 2.
4. Процесс повторяется, пока обе таблицы не будут пусты.
5. Как только таблицы пусты, подать следующий входной набор.

78. Синтез установочной последовательности методом Рота.

Основной причиной связанной с громоздкостью метода ТФН, является необходимость рассмотрения одновременно всех неисправностей схемы. Это приводит к очень большим размерностям таблиц неисправности. Рот предложил строить тест для одной неисправности. Учитывая, что количество неисправностей всегда конечно и что тест всегда обнаруживает несколько неисправностей, процедура предложенная Ротом такова:

Из списка всех неисправностей выбирается любая и для неё строится тест, так как этот тест обнаруживает еще ряд неисправностей, то все они вычеркиваются из таблиц неисправности, затем из оставшегося списка выбирается любая и так далее. В результате:

1) Все неисправности обнаружены и контролирующий тест построен
2) Осталась часть неисправностей, для которой тест построить не удалось

Построение теста для конкретной неисправности связано с исчислением D – кубов Рота. D – куб Рота строится по таблице истинности отдельного элемента. Таблица истинности делится на две части. В часть компонуется наборы связанные одинаковыми выходными сигналами. Осуществляется пересечение векторов внутри каждой части.