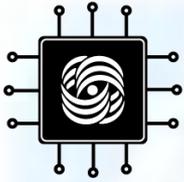


НАДЁЖНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лекция 8: *Надёжность системы*

ВМиК МГУ им. М.В. Ломоносова,
Кафедра АСВК, Лаборатория Вычислительных Комплексов
Ассистент Волканов Д.Ю.



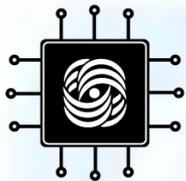
О чём этот курс

- Концепции и связи
- Аналитические модели и средства поддержки
- Методы для увеличения надёжности ПО:
 - Верификация
 - Статический анализ
 - *Тестирование*
 - **Расчёт надёжности**
 - *Анализ статистики ошибок*
 - *Безопасность*



План лекции

- Надёжность последовательных и параллельных систем
- Блок-схема расчёта надёжности
- Анализ эксплуатационной безопасности
- Анализ типов и последствий отказов
- Анализ дерева неисправностей

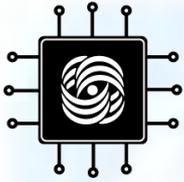


Эпиграф

- *В жизни нет гарантий, существуют
одни вероятности*

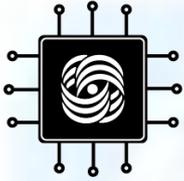
Том Клэнси





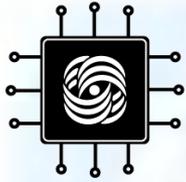
Что такое надежность?

- Надежность - вероятность того, что программная система будет работать по спецификации.
- Дефект – код, приводящий к ошибке.
- Ошибка – проявление дефекта.
- Ошибки приводят к отказам – неверной работе программы.



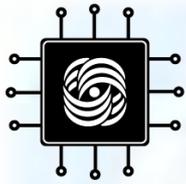
Модели надёжности

- Граф надёжности (Reliability Graph)
- Дерево неисправностей (Fault Tree model)
- Блок-схема расчёта надёжности (Reliability Block Diagram)



Системная Надёжность (1)

- Система состоит из модулей
- Модуль состоит из компонент
- Компоненты могут иметь
 - Разную надёжность
 - Разную зависимость между ними
- Надёжность системы – это функция надёжности компонент и зависимости между компонентами

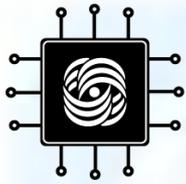


Надежность последовательных систем

- Система состоит из n компонент, соединенных друг с другом
- Отказ каждого компонента приводит к отказу всей системы



- Надежность системы меньше надежности ее компонентов (т.к. $R_k \leq 1$).



Суммирование надежностей

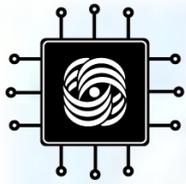
- Надежность системы может быть выражена через надежности ее компонентов, если они отказывают независимо.
- Для систем из отдельных компонентов:

$$R = \prod_{k=1}^{Q_p} R_k$$

Q_p число компонентов

R_k надежность компонента

- Надежности компонентов (R_k) должны быть заданы в одном интервале.



Суммирование надёжностей

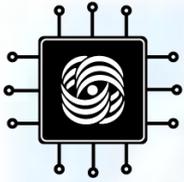
- Используя связь между надёжностью и частотой ошибок

$$\lambda = \frac{-\ln R}{\tau} \quad \text{or} \quad R = e^{-\lambda\tau}$$

- Получаем:

$$\lambda = \sum_{k=1}^{Q_p} \lambda_k$$

- Т.е. общая частота есть сумма частот отдельных компонентов



Пример системы

- Система из 4 независимых КОМПОНЕНТОВ

- $R_1 = 0.95$

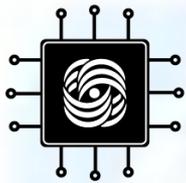
- $R_2 = 0.87$

- $R_3 = 0.82$

- $R_4 = 0.73$

$$R_{system} = 0.95 \times 0.87 \times 0.82 \times 0.73 = 0.4947$$

- Надежность системы меньше надежности каждого из компонентов

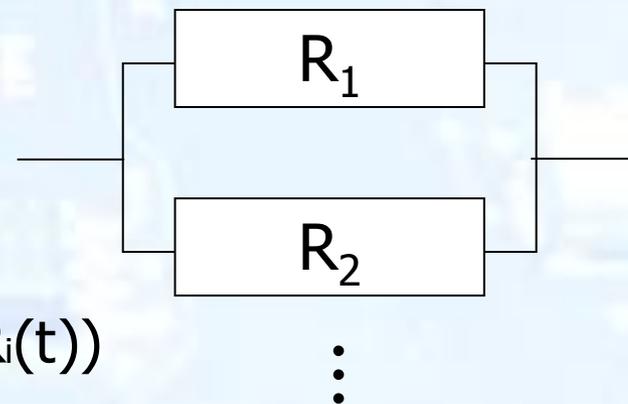


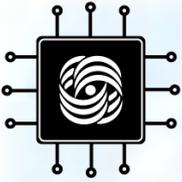
Надежность параллельных систем

- Система состоит из n компонентов, работающих параллельно
- Только отказ всех компонентов приводит к отказу всей системы.

$$\text{Unreliability } F_p(t) = \prod_{i=1}^n F_i(t)$$

$$\text{Reliability } R_p(t) = 1 - \prod_{i=1}^n F_i(t) = 1 - \prod_{i=1}^n (1 - R_i(t))$$





Пример параллельной системы

- Система состоит из 4 параллельно работающих компонентов

- $R_1 = 0.95$

- $R_2 = 0.87$

- $R_3 = 0.82$

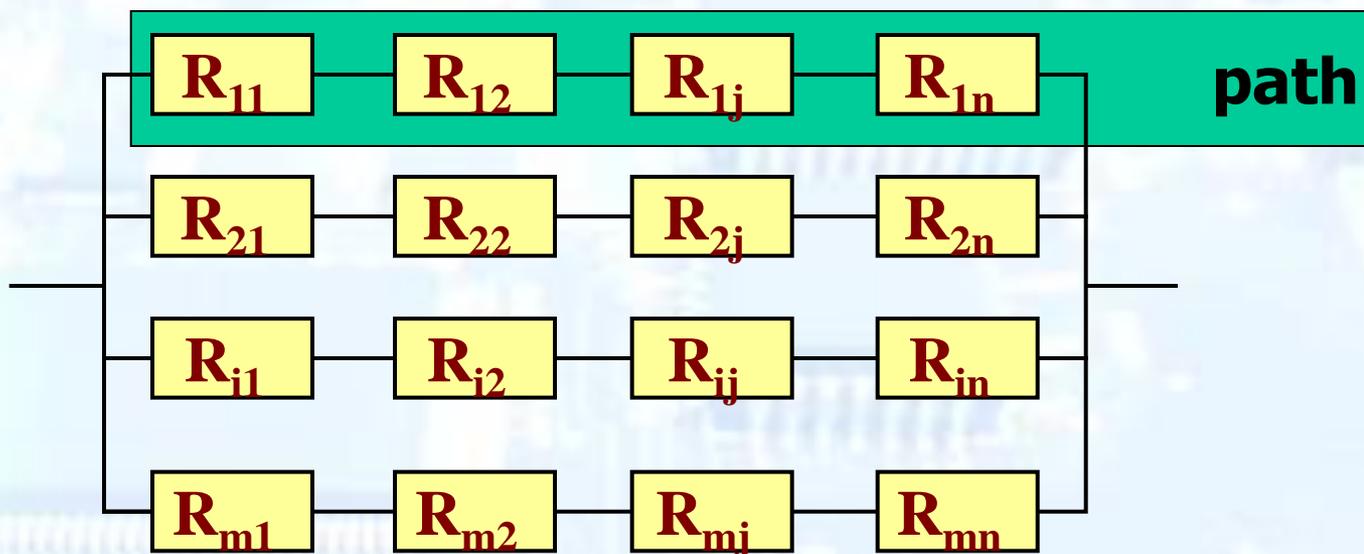
- $R_4 = 0.73$

$$R_{system} = 1 - ((1 - 0.95) \times (1 - 0.87) \times (1 - 0.82) \times (1 - 0.73)) = 0.9996$$

- Надежность системы больше, чем надежности отдельных компонентов.

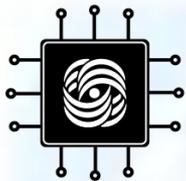


Параллельно-последовательные системы

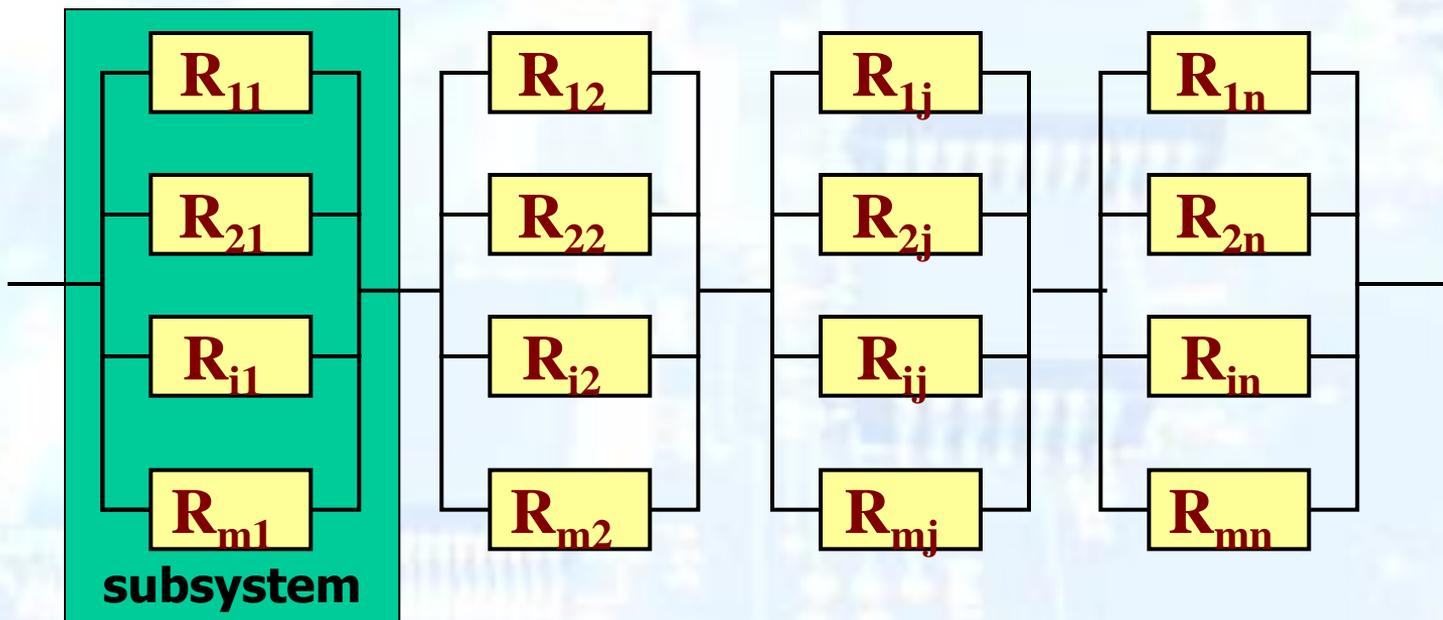


$$\text{Pathreliability } R_i = \prod_{j=1}^n R_{ij}$$

$$\text{Systemreliability } R = 1 - \prod_{i=1}^m (1 - R_i) = 1 - \prod_{i=1}^m \left(1 - \prod_{j=1}^n R_{ij}\right)$$

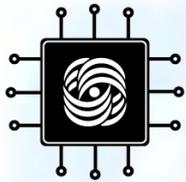


Параллельно-последовательные системы



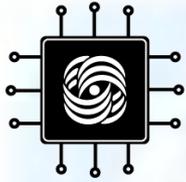
$$\text{Subsystem reliability } R_j = 1 - \prod_{i=1}^m (1 - R_{ij})$$

$$\text{System reliability } R = \prod_{j=1}^n R_j = \prod_{j=1}^n \left[1 - \prod_{i=1}^m (1 - R_{ij}) \right]$$



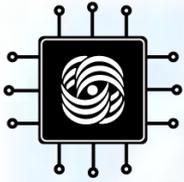
Другие архитектуры

- Однополосный мост
- Двуполосный мост



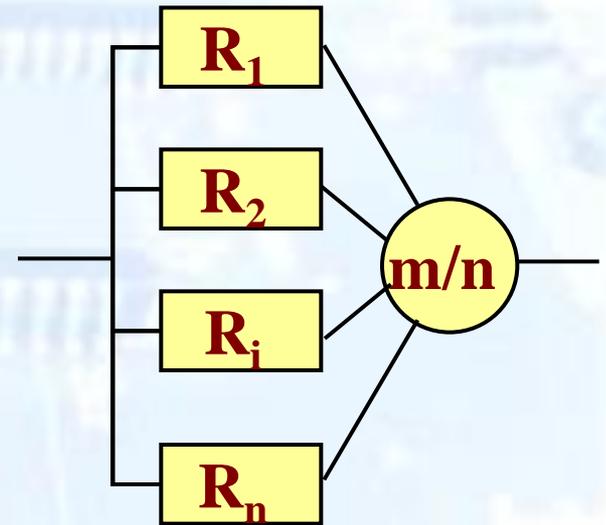
Горячее резервирование

- Использует параллельность.
- Все компоненты работают одновременно
- Каждый компонент достаточен для корректной работы
- Только один компонент необходим для корректной работы
- Каждый компонент удовлетворяет требованиям к надежности
- Система отказывает, если отказывают все компоненты.



Система m – из – n

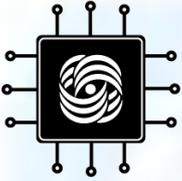
- Система из n компонентов
- Для корректной работы нужно не меньше m компонент ($m \leq n$).
 - $m=n$: последовательная
 - $m=1$: параллельная
- Например, самолет с 4 моторами может лететь при отказе двух моторов



Предположение: Все компоненты имеют одинаковую надежность

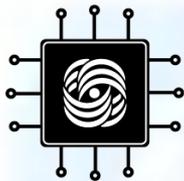
$$R_{\text{sys}}(t) = 1 - \sum_{i=0}^{m-1} \binom{n}{i} R(t)^i (1 - R(t))^{n-i}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

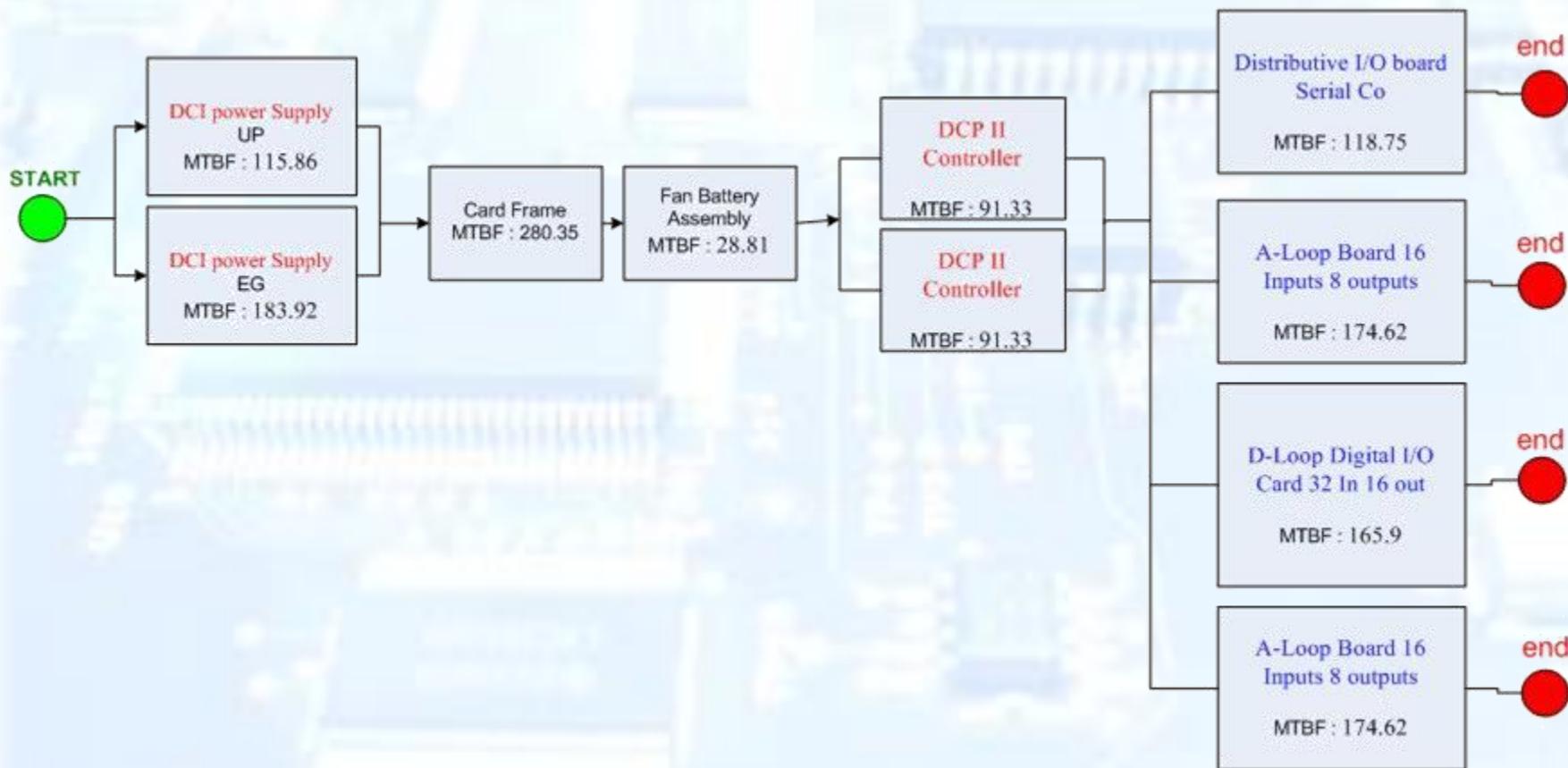


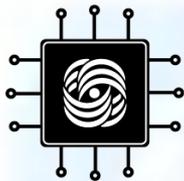
Блок-схема надежности (RBD)

- RBD показывает, как соединены компоненты системы для обеспечения надежности
- Самые частые конфигурации – последовательные и параллельные компоненты.
- В последовательной конфигурации отказ каждого компонента приводит к отказу системы. Общая надежность меньше надежности отдельных компонентов.
- В параллельной конфигурации отказ всех компонентов приводит к отказу системы. Общая надежность больше надежности отдельных компонентов.
- Система содержит некую комбинацию этих конфигураций
- Анализ диаграмм необходим для вычисления надежности систем.

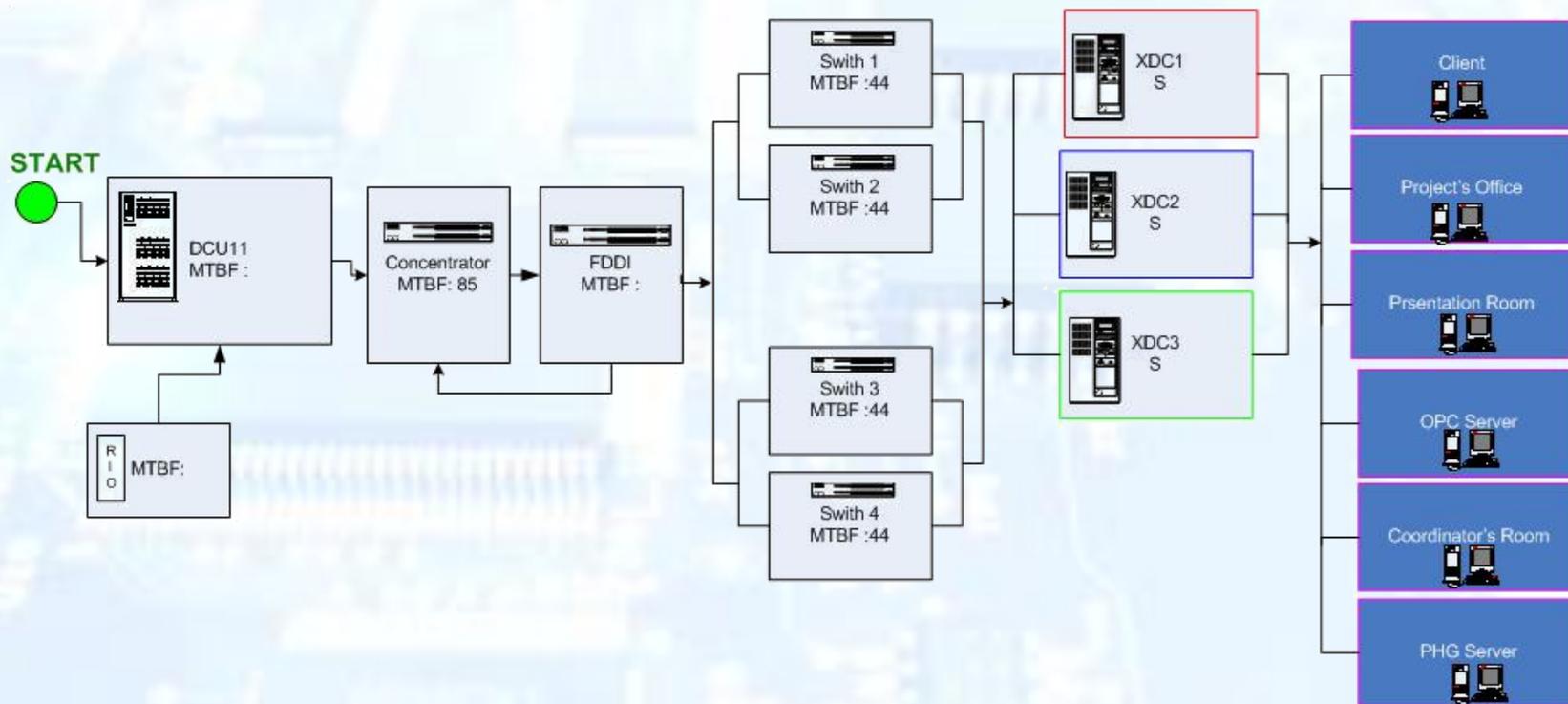


RBD: пример /1





RBD: пример /2





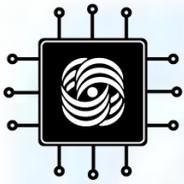
RBD: когда и как?

- Когда использовать RBD?
 - Когда нужно оценить надежность сложной системы и найти в системе уязвимые места
- Как использовать RBD?
 - Нарисовать схему
 - Найти надежность системы
 - Вычислить другие характеристики (доступность, время работы etc)
- Есть автоматизированные средства создания и анализа диаграмм, интегрированные с другими методами, например, с FTA.



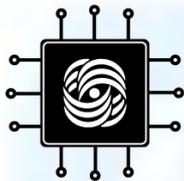
RBD: плюсы и минусы

- RBD – самый простой метод визуализации характеристик надежности.
- Преимущества:
 - Помогает найти цели по надежности
 - Оценивает влияние отказа компонента на работу системы
 - Позволяет простой анализ различных сценариев работы
 - Подсчитывает MTBF системы
 - Уменьшает стоимость исправления проблем в больших системах
 - Позволяет анализировать разные конфигурации
 - Указывает на потенциальные проблемы в системе
 - Определяет чувствительность системы к отказам компонентов
- Недостаток в том, что сложные конфигурации (ожидание, ветвление, распределение нагрузки и.т.д.) не могут быть описаны RBD.



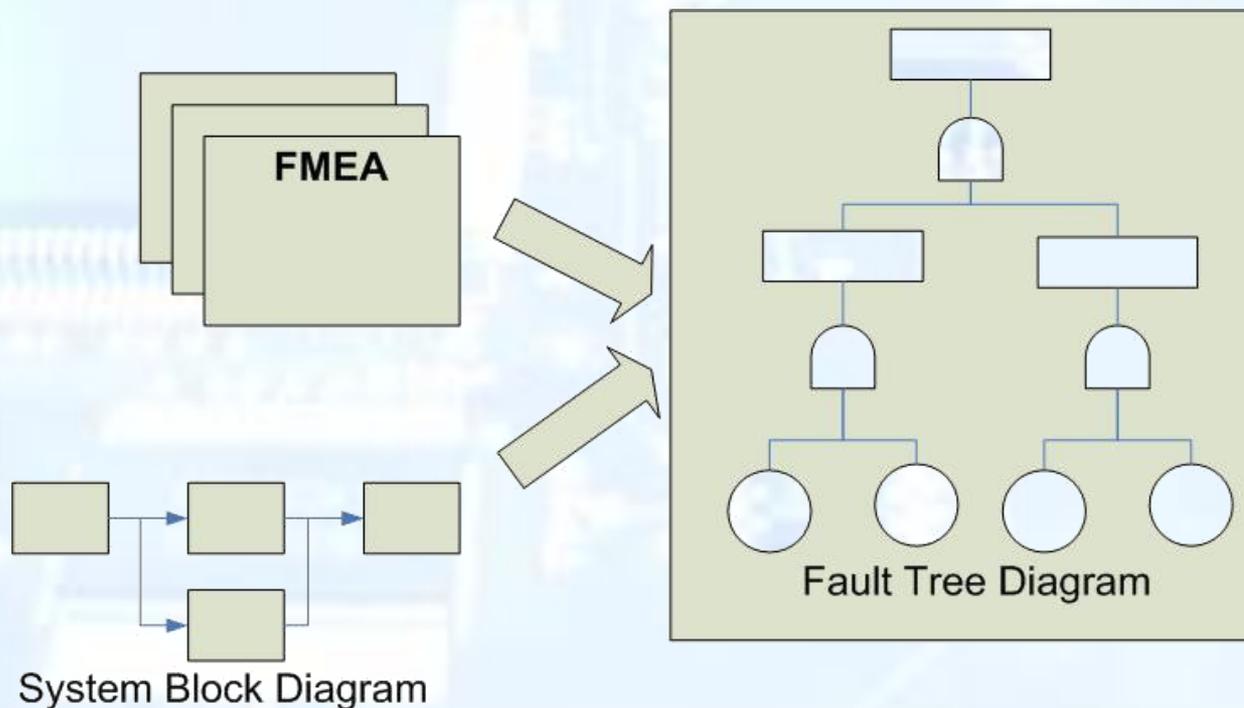
Дерево неисправностей (FTA)

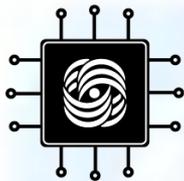
- Дерево неисправностей – графическое представление основных ошибок и отказов в системе, их причин и способов их устранения. Деревья позволяют находить уязвимые места при проектировании новой системы или анализе существующей. Они также помогают находить методы решения проблем.
- Дерево неисправностей можно рассматривать как графическую модель поведения системы, ведущего к нежелательному событию. Поведение определяется событиями и условиями и записывается логическим выражением.
- Анализ деревьев неисправностей полезен как при проектировании новых систем и программных продуктов, так и при поиске недостатков в существующих. Они помогают найти корень проблемы и исправить ее.



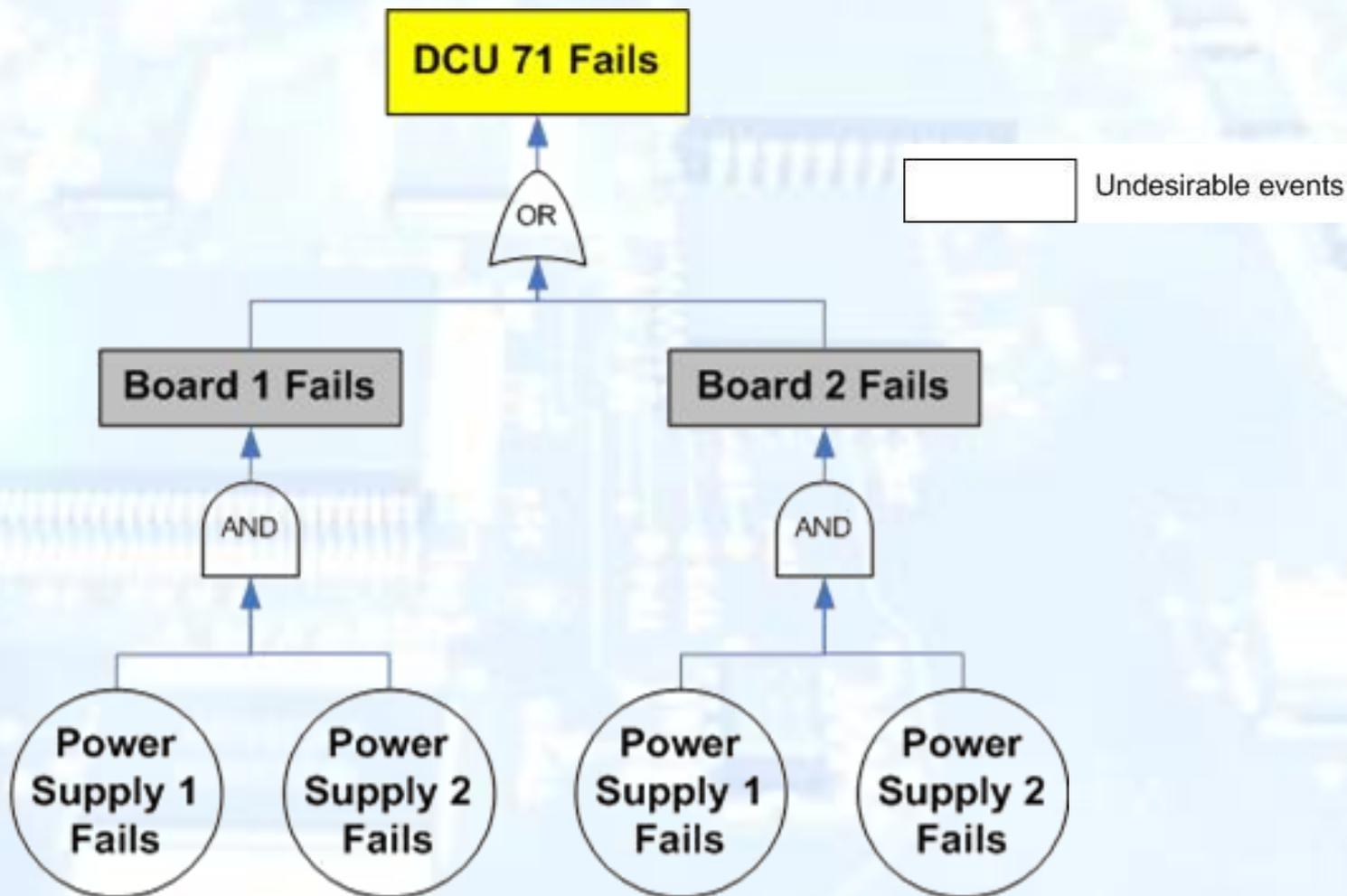
Шаги в FTA

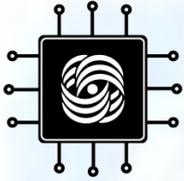
- FMEA определяет условия, приводящие к нежелательным событиям. Затем определяется, какие локальные ошибки могут привести к отказу системы. FMEA отвечает на вопрос, “Что может пойти не так?” даже если спецификация выполнена. **Для работы FTA необходимо иметь FMEA.**





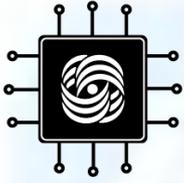
FTA: пример





FTA: плюсы и минусы

- **Плюсы:** FTA дают осмысленные данные, позволяющие оценить и улучшить надежность системы. Они также дают оценку необходимости и эффективности резервирования.
- **Минусы:** Ограничение FTA в том, что нежелательное событие необходимо предвидеть и знать основные источники проблемы. Это может быть очень долго и дорого. Успех зависит от способностей аналитика, работающего над этой задачей.



Спасибо за внимание!