Второй док

<https://drive.google.com/file/d/1VwtbMwuN7A01_hlE81MGnkDSu0ssXl0p/view>

# Словарь сокращений

* ВВМ — виртуальная вычислительная машина
* ВПл — вычислительная платформа
* ВС — вычислительная система
* ВсС — встраиваемая вычислительная система
* ИУС — информационно-управляющая система
* КФС — киберфизическая система
* MoC (англ.) — модель вычисления
* РИУС — распределённая информационно-управляющая система
* СнК — система на кристалле

# 1. Понятие ВС и вычислительного процесса (ВП). Классификация ВС.

**ВЫЧИСЛИ́ТЕЛЬНАЯ СИСТЕ́МА** — совокупность *аппаратно-программных* средств, образующих единую среду, предназначенную для решения задач обработки информации (вычислений).

**Вычислительный процесс** – процесс преобразования данных вычислительным устройством (вычислителем, вычислительной машиной) в соответствии с заданной функциональностью. (Может выполняться однократно или циклически).

**Вычислитель** – функционально завершенное устройство, позволяющее осуществлять один или более вычислительных процессов. Процесс преобразования данных описывается алгоритмом на алгоритмическом языке.

Из известных общих классификаций ВС для сегодняшнего состояния вычислительной техники видится наиболее удачной классификация Дэвида Паттерсона, в соответствии с которой выделяются три категории вычислительных систем:

* настольные компьютеры (ПК – рабочая станция / интеллектуальный терминал);
* серверы (ВС коллективного пользования);
* встраиваемые системы (все прочие ВС).

# 2. Организация ВП и проектирование ВС.

В проектировании ВсС известна высокая значимость и сложность инструментального обеспечения. Выбор, как архитектуры создаваемой системы, так и технологии ее проектирования и отладки теснейшим образом переплетаются в деятельности разработчика, оказывая сильное взаимное влияние. Важным является вопрос разделения инструментальных компонент на резидентные и внешние составляющие. Также большое значение имеет выделение элементов подготовительной и исполнительной частей вычислительного процесса. Все это непосредственно влияет на архитектуру и реализацию целевой вычислительной системы. Объединение потоков проектирования целевого продукта и его инструментального обеспечения на ранней стадии работ позволяет значительно повысить качество создаваемой ВсС и улучшить все основные характеристики процесса проектирования. Таким образом, процесс проектирования ВсС как объекта, решающего конечную задачу, предлагается рассматривать как **организацию вычислительного процесса** в пространстве и времени в оговоренных техническим заданием ограничениях.

Процедура организации вычислительного процесса обладает различной функциональной сложностью в зависимости от класса и количества решаемых задач, режимов обработки данных, топологии системы обработки данных. При обработке данных с помощью ЭВМ в зависимости от конкретного применения информационной технологии, а значит, и решаемых задач различают три основных режима: **пакетный, разделения времени, реального времени**.

При **пакетном режиме** обработки задания (задачи), а точнее, программы с соответствующими исходными данными, накапливаются на дисковой памяти ЭВМ, образуя "пакет". Обработка заданий осуществляется в виде их непрерывного потока. Размещенные на диске задания образуют входную очередь, из которой они выбираются автоматически, последовательно или по установленным приоритетам. Входные очереди могут пополняться в произвольные моменты времени. Такой режим позволяет максимально загрузить ЭВМ, так как простои между заданиями отсутствуют, однако при получении решения возникают задержки из-за того, что задание некоторое время простаивает в очереди.

**Режим разделения времени** реализуется путем выделения для выполнения заданий определенных интервалов времени, называемых квантами. Предназначенные для обработки в этом режиме задания находятся в оперативной памяти ЭВМ одновременно. В течение одного кванта обрабатывается одно задание, затем выполнение первого задания приостанавливается с запоминанием полученных промежуточных результатов и номера следующего шага программы, а в следующий квант обрабатывается второе задание и т.д. Задание при этом режиме находится все время в оперативной памяти вплоть до завершения его обработки. При большом числе одновременно поступающих на обработку заданий можно для более эффективного использования оперативной памяти временно перемещать во внешнюю память только что обрабатывавшееся задание до следующего его кванта. В режиме разделения времени возможна также реализация диалоговых операций, обеспечивающих непосредственный контакт человека с вычислительной системой.

**Режим реального времени** используется при обработке данных в информационных технологиях, предназначенных для управления физическими процессами. В таких системах информационная технология должна обладать высокой скоростью реакции, чтобы успеть за короткий промежуток времени (лучше мгновенно!) обработать поступившие данные и использовать полученные результаты для управления процессом. Поскольку в технологической системе управления потоки данных имеют случайный характер, вычислительная система всегда должна быть готова получать входные сигналы и обрабатывать их. Повторить поступившие данные невозможно, поэтому потеря их недопустима.

В ЭВМ используют также режимы, называемые ***однопрограммными и мультипрограммными***.

***Однопрограммный режим*** - режим, при котором выполняется не более одной независимой программы. При таком режиме работы ЭВМ решение задачи начинается с загрузки программы в ОП, после чего ЭВМ последовательно выполняет команды программы. При этом в каждый момент времени работает только одно ее устройство, в то время как остальные простаивают в ожидании окончания ранее начатого действия.

В ***мультипрограммном режиме*** в основной памяти одновременно находится несколько программ, загруженных для выполнения. Центральный процессор в каждый момент времени может выполнять лишь одну из программ. Таким образом, параллельно выполняемые программы конкурируют между собой за обладание ресурсами вычислительной системы и в первую очередь за время центрального процессора. Задача, обладающая в текущий момент центральным процессором выполняется до тех пор, пока не окажется в состоянии ожидания какого-либо события (например, завершения операции ввода-вывода) или не будет прервана по каким-либо причинам операционной системой. Остальные задачи находятся в состоянии готовности использования центрального процессора, либо ожидания какого-либо события (завершения операции ввода-вывода, истечения заданного интервала времени, завершения выполнения какой-либо программы и т.д.). После перевода активной задачи в состояние ожидания выбирается одна из готовых для выполнения задач, которая становится активной.

В режиме разделения времени используется вариант мультипрограммного режима.

**Проектирование ВС**

— процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части *(ISO 24765)*. Результатом проектирования является **прое́кт** — целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы.

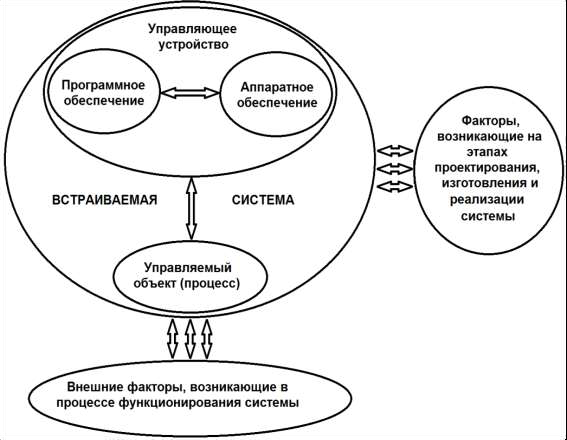
— процесс разработки конструкторской и программной документации, необходимой для производства и эксплуатации системы. Цель проектирования, назначение ВС, исходные данные и технические требования устанавливаются ТЗ. ТЗ должно содержать:

* Перечень функций системы
* Состав и характеристики источников и приемников информации, их территориальное размещение
* Требуемые значения характеристик ВС (производительность, стоимость, надежность, емкость памяти, габариты, энергопотребление)
* Необходимые режимы обработки данных
* Условия эксплуатации и другие факторы

# 3. Понятие ВсС, СнК, классификация.

## ВсС

**Встраиваемые вычислительные системы** (ВсС) - специализированные (заказные) микропроцессорные системы, непосредственно взаимодействующие с объектом контроля или управления и объединенные с ним конструктивно.



Характеристика реактивных систем реального времени:

* реагируют на состояние внешней среды;
* постоянный цикл взаимодействия со средой;
* в идеале, выполняют бесконечный целевой алгоритм;
* должны учитывать внешние временные ограничения (реальное время).

Бонус:

* “Встроенной системой можно считать любую вычислительную систему, которая не является ПК, портативным компьютером (laptop) или большим универсальным компьютером (mainframe computer)”
* “Устройство, которое включает в себя программируемый компьютер, но не является при этом компьютером общего назначения”
* “Система обработки информации, встроенная в какой-либо продукт”

## СнК

**Систе́ма на криста́лле** (System on Chip, SoC, СнК)  
[ВсС, выполняемая по технологии «система на кристалле»]

— в микроэлектронике — электронная схема, выполняющая функции целого устройства (например, компьютера) и размещенная на одной интегральной схеме.

— система, построенная на едином кристалле, в которой интегрируются такие элементы, как процессор (процессоры, в том числе специализированные), некоторый объем памяти, ряд периферийных устройств и специализированных вычислительных блоков, и их соединения.

СнК часто используются в качестве компонентов ВсС и в предельных случаях фактически ими же и являются

Характеристика СнК:

* сборка «готовых компонентов», зачастую приобретенных у сторонних производителей («интеллектуальная собственность»)
* иерархия «черных ящиков»
* проектирование и верификация выполняются больше на системном уровне, чем на логическом
* акцент на взаимодействие компонент
* большая важность программного обеспечения

Преимущества "систем на кристалле" перед классическими "системами на печатной плате":

1. ***Миниатюризация***  
   Как правило, устройство, созданное на баз SoC, состоит из одной (максимум 2-х) СБИС и ограниченного набора дискретных компонент, которые по технологическим причинам не могут быть интегрированы внутрь ИС.
2. ***Снижение потребляемой мощности***  
   СБИС типа "система на кристалле" изготавливаются по технологии "глубокого субмикрона" (DSM — Deep Submicron) 0.35 мкм и ниже, что позволяет снизить напряжение питания и, как следствие, существенно уменьшить потребляемую мощность.
3. ***Повышение надежности***Объединение нескольких компонент (IP-блоков) на одной пластине кремния позволяет существенно уменьшить число паяных соединений.
4. ***Снижение стоимости*** для больших партий  
   ВсС, выполненных на базе СнК, при налаженном производстве являются гораздо более дешёвыми решениями, чем обычные системы на платах.
5. ***Упрощение монтажа***  
   При использовании технологии СнК требуется устанавливать меньше корпусов на плату — как правило, сам вычислительный элемент.

## Классификация

На сегодняшний день отсутствует четкое определение класса ВсС, и проявляется это в проблемах классификации как самих ВС, так и архитектурных парадигм в их проектировании. ВС можно классифицировать по многим признакам:

* характер использования
* степень интеграции
* функциональный признак
* принцип комплексирования (полностью готовые, блочные, заказные, полузаказные или смешанные)
* сложность системы (большие, средние, малые)
* топология системы (сосредоточенные, распределенные)
* тип ВС, являющейся основой ВсС (одно- и многопроцессорные, гомогенные и гетерогенные, сильно и слабо связанные, использующие ОС и не использующие)
* особенность реализации реального масштаба времени (мягкое и жесткое реальное время)
* конструкция (моноблочные, модульные, встроенные, расширяемые и нерасширяемые, обслуживаемые и необслуживаемые)
* реализуемая для ВсС надежность, безопасность, информационная защищенность и т.д.
* другие функциональные и нефункциональные характеристики

и т. д.

### По степени интеграции с объектом управления

1. Информационно-управляющие системы (ИУС)
2. Распределённые ИУС
3. ВсС
4. Сетевые ВсС
5. Киберфизические системы

Порядок элементов в списке выше соответствует хронологии развития ВсС.

### По функциональному признаку

Разделить ВсС по назначению можно только очень условно. Попытка такого деления представлена ниже:

1. Системы автоматического управления (САУ);
2. Измерительные системы и системы сбора информации с датчиков (приборные, характерны ярко выраженные измерительные функции наряду с управлением);
3. Информационные системы “запрос-ответ” реального времени (платежные системы, резервирование билетов и т.д.) Они занимают пограничное положение с информационными системами общего назначения;
4. Цифровые системы передачи данных (телекоммуникационные системы);
5. Сложные иерархические системы реального времени (обеспечивают контроль и управление сложными, в том числе, пространственно-распределенными объектами);
6. Системы управления подвижными объектами;
7. Подсистемы ВС общего назначения;
8. Мультимедийные системы.

Стр 21-24 из:  
<https://vk.com/doc81390383_533235091?hash=26407b68fd3e0f8223&dl=abf9b1d0685d67aec0>

# 4. Понятие киберфизической системы (КФС), место и роль информатики и ВТ в проектировании КФС.

**Киберфизические системы** (Cyber-Physical **System**, CPS) — это **системы**, состоящие из различных природных объектов, искусственных подсистем и управляющих контроллеров, позволяющих представить такое образование как единое целое. В CPS обеспечивается тесная связь и координация между вычислительными и физическими ресурсами.

<http://cyberphysicalsystems.org/> **Киберфизическая система** (Cyber-Physical System, CPS) — информационно-технологическая концепция, подразумевающая интеграцию вычислительных ресурсов в физические сущности любого вида, включая биологические и рукотворные объекты.

Термин «**киберфизические системы**», подразумевает интеграцию вычислений с физическими процессами. Понятие отражает ожидающийся в ближайшем будущем качественный переход в восприятии ВсС и методов их проектирования. Суть CPS состоит в том, что проектирование объекта управления и системы управления для этого объекта должны выполняться в едином ключе, в едином комплексе, тесно взаимодействующих инструментальных средств. Кроме того, концепция CPS, как правило, предполагает объединение в сеть с другими устройствами, тесное взаимодействие и обмен информацией с этими устройствами.

Примеры:Умные сети электроснабжения,Беспило́тный автомоби́ль  
Это могут быть электронно-механические системы, гидравлические, биологические и т.д.

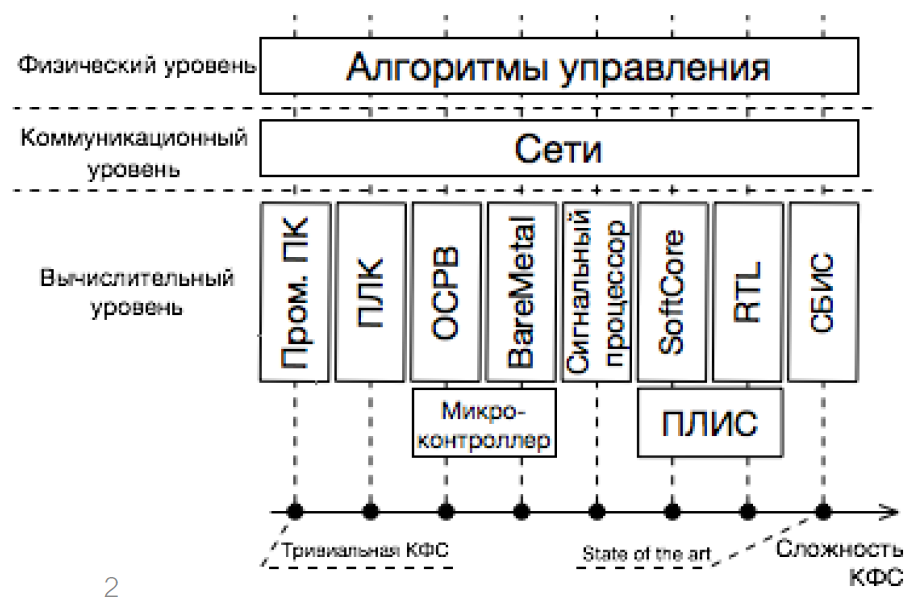
Общие проблемы проектирования КФС с позиции инфо-коммуникационной составляющей:

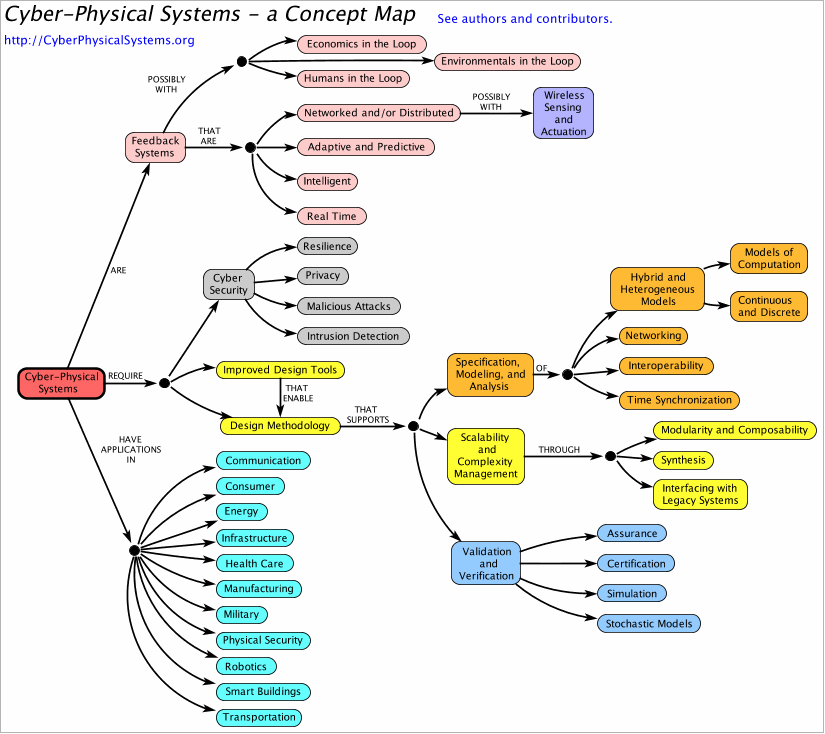
* Необходимость большого объема вычислений здесь и сейчас
* Жесткие ограничения на энергопотребление / размер / производительность

Сложные КФС требуют согласованности уровней организации КФС, специализации инфо коммуникационной составляющей для алгоритма управления

Рассогласованность уровней КФС и работы специалистов разных профилей приводит к:

* падению качества и скорости проектирования
* падению качества результата





# 5. ВС реального времени

## Определение

Вычислительная **система реального времени** — это система, которая должна реагировать на события во внешней по отношению к системе среде или воздействовать на среду *в рамках требуемых временных ограничений*.

Другими словами, обработка информации системой должна производиться за *определенный конечный период времени*, чтобы поддерживать постоянное и своевременное взаимодействие со средой.

Принципиальное отличие информационных систем от систем реального времени, к которым, в большинстве случаев, относятся ВсС, состоит в трактовке параметра «реакция вход-выход»: «The right answer late is wrong» («Правильный ответ поздно – неправильный»).

## Применение

С развитием технологий системы реального времени нашли применения в самых различных областях. Особенно широко СРВ применяются в промышленности, включая системы управления технологическими процессами, системы промышленной автоматики, SCADA-системы, испытательное и измерительное оборудование, робототехнику. Применения в медицине включают в себя томографию, оборудование для радиотерапии, прикроватное мониторирование.

## Примеры

* система обработки аудио- и видеопотоков при трансляции в прямом эфире;
* интерактивная компьютерная игра.
* бортовая система управления космического аппарата;

# 6. Понятие архитектуры и микроархитектуры в вычислительной технике.

**Архитектура ВС** - совокупность структурных и функциональных элементов ВС некоторого уровня детализации, адекватно отображающая проектируемую систему для данного уровня рассмотрения.

**Архитектура системы** - принципиальная организация системы, воплощенная в ее элементах, их взаимоотношениях друг с другом и со средой, а также принципы, направляющие ее проектирование и эволюцию.

**Микроархитектура** - это способ реализации данной архитектуры набора команд (ISA) в конкретном процессоре. Данный ISA может быть реализован с разными микроархитектурами; реализации могут отличаться из-за разных целей данного проекта или из-за изменений в технологии.

**Микроархитектура** - функционально завершенный набор вычислительных компонент (микропроцессор, периферия, программируемая логика, память).

**Компьютерная архитектура** - это сочетание микроархитектуры и архитектуры набора команд.

# 7. Уровневая модель ВС.

источник:<https://cyberleninka.ru/article/n/testirovanie-i-otladka-vstraivaemyh-vychislitelnyh-sistem-na-osnove-urovnevyh-modeley>

Уровневое представление вычислительных систем является широко используемым и эффективным способом описания их вертикальной организации, отражающим набор используемых уровней абстракции и соответствующих технологий разработки, в том числе из разных областей вычислительной техники. Значительными выразительными возможностями для моделирования уровневой организации вычислительных систем обладает архитектурный стиль «модель-процесс-вычислитель» (МПВ) [12], что определило его использование в предлагаемой методике тестирования и отладки в качестве способа описания комплексных проектов ВсС.

Источник:презентация пенского architecture-styles-slides из репы на гите





Рассматривая архитектуру ВсС через призму целевой функциональности, можно говорить о различных уровнях детализации представления, о различной направленности представления (для категорий специалистов), о различной степени оптимальности реализации в соотношении с технологической сложностью. ***Более высокоуровневым, доступным для восприятия (прозрачным) и простым технологически представляется способ подачи архитектуры ВсС как простой иерархии виртуальных машин.*** При этом считается, что виртуальная машина – вычислитель (computer), полученный в результате виртуализации базовой вычислительной платформы. Но требование минимизации ресурсов вступает в противоречие с подобной моделью, заставляя сокращать число уровней иерархии и переходить к так называемым "плоским" моделям реализации. Иерархическое представление ВсС в терминах виртуальных машин является очень важным и мощным инструментом проектирования. Важнейшее свойство такого представления состоит в возможности достигать сокращения трудоёмкости проектирования и повышения степени повторного использования при условии выполнения других ресурсных ограничений проекта. Кроме того, такое представление полезно при обучении и формировании специалистов в области ВсС, так как оно позволяет демонстрировать воспринимаемый человеком образ системы, заставляет оперировать в явном виде различными вычислительными моделями, согласовывать их друг с другом.

Переход от высокоуровневого абстрактного представления вычислительного процесса к уровню физической реализации так называемого "не вычислительного базиса" (границей можно считать вентильный или транзисторный уровень) в силу сложности задачи требует большого числа промежуточных уровней и представлений. Эти уровни необходимы, прежде всего, разработчику для борьбы со сложностью задачи. На практике чаще всего выделение и реализация таких уровней в вычислительной системе выполняется различными коллективами разработчиков и на разных этапах проектирования. Причем в большинстве проектов значительная (основная) часть работы присутствует в повторно используемых компонентах, таких как процессоры, операционные системы, различные API, коммуникационные протоколы, трансляторы и т.д.



# 8. Задачи этапа высокоуровневого (архитектурного) проектирования ВС.

На сегодняшний день **задачи** этапа **высокоуровневого проектирования (SLD - System Level Design)** ВсС группируются в шесть крупных блоков, покрывая фазу проектирования, и не затрагивая фазу реализации:

1. Концепция решения целевой задачи, исходные спецификации.
2. Организация вычислительного процесса (модели вычислений – англ. MоC).
3. Генерация архитектуры и микроархитектуры.
4. Оценка и выбор архитектурных решений.
5. Верификация архитектурных решений.
6. Создание спецификаций для этапа реализации

Суть SLD кратко и доступно можно охарактеризовать известной поговоркой: «семь раз отмерь, один раз отрежь». При системном подходе предполагаются некоторые дополнительные усилия перед началом самой «работы» (точнее, перед тем, что обычно понимают под этим словом: перед созданием аппаратуры, написанием программ, и т.д.), направленные на то, чтобы впоследствии не пришлось всё переделывать из-за явных ошибок или недостаточной эффективности конечного изделия.

# 9. Аппаратная и программная реализация в вычислительной технике.

<https://www.youtube.com/watch?v=UMRNfWSwmPo>

Мб это?

Теоретически алгоритмы любой сложности можно отображать на комбинационных схемах и триггерах, возможно – с обратными связями, таким образом, чтобы на вход подавались исходные данные, и на выходе через какое-то время появлялся результат. Это так называемая потоковая обработка данных, когда данные обрабатываются вычислительными узлами сразу же, как только поступают на их входы. Но практически реализация такого рода применяется только в некоторых случаях, когда это действительно оправдано. Это происходит из-за трудоёмкости проектирования подобных схем, необходимости использования большого количества транзисторов на кристалле для реализации всех её базовых элементов, и отсутствия гибкости, т.е. возможности изменять функциональность в широких масштабах для решения различных задач.

Более распространёнными в современной вычислительной технике являются ВС с использованием фон-Неймановской архитектуры. Простейшие ВС, функционирующие на этих принципах, можно представить, как небольшую фиксированную универсальную схему, которая способна выполнять несколько простейших действий (операции сложения, записи в память и чтения из памяти, логические операции и т.д.). Выбор конкретного действия из всех возможных определяется поданной на вход управляющей последовательностью. Значения управляющей последовательности в заданном порядке выгружаются из памяти, оттуда же берутся также состояния входов схемы и туда записываются результаты выполнения действий с выходов схемы. Координация вычислений и всего функционирования схемы производится управляющей логикой, контроллером. Набор управляющих последовательностей, изменяющих состояние схемы, называется программой.

Мб, это как в кристаллах было и можно свести к IP-ядрам?

#### **Soft IP**

Следующий блок, который мы рассмотрим — это процессорное ядро. Собственно, на его месте может быть практически любая цифровая схема.

Основное отличие разработки цифровых схем от разработки аналоговых состоит в том, что цифровые схемы обычно пишутся на специальных языках высокого уровня — Verilog или VHDL, а дальше специальный САПР синтезирует из этого кода электрическую схему и топологию. Так вот, до момента синтеза код не привязан к конкретной технологии, и один и тот же блок можно произвести на разных фабриках (или зашить в ПЛИС) или на разных вариантах одной и той же технологии (например, оптимизированных под высокую скорость или низкое энергопотребление). Этим, собственно, и пользуются компании-разработчики, такие как, например, ARM. Такие цифровые блоки, не привязанные к определенной технологии, называются Soft IP.

**Жесткие ядра**

Жесткие ядра определяются как IP-ядра, которые не могут быть модифицированы и, таким образом, являются «жесткими», аналогично этимологии аппаратного и программного обеспечения. В силу своего низкоуровневого представления, жесткие ядра обеспечивают лучшую предсказуемость производительности чипа в сроки выполнения работ и области.

Аналоговая и смешанная логика обычно определяются как физическое описание более низкого уровня.

Такие ядра, как аналоговые, так и цифровые, называются «жесткими ядрами» (или жесткими макросами), поскольку разработчики чипов не могут существенно изменить функцию приложения ядра. Схемы транзисторов должны соответствовать правилам проектирования процессов целевой литейной фабрики, и, следовательно, жесткие ядра, доставленные для одной литейной промышленности, не могут быть легко перенесены на другой процесс или литейную фабрику.

# 10. Понятие виртуальной машины.

ВВМ (**виртуальная вычислительная машина**) – техническое решение, реализующее вычислительный процесс. Разделение ВС (вычислительной системы) на механизмы и машины возможно относительно выделенных (в качестве самостоятельных) вычислительных процессов.

**Виртуальная машина** – вычислитель (computer), полученный в результате виртуализации базовой вычислительной платформы.

**Виртуализация** – абстрактное (модельное) представление вычислительных ресурсов программно-аппаратной платформы.

Сама по себе модель того или иного вычислителя не подразумевает реализации как таковой, однако каждая виртуальная машина имеет конкретную реализацию, как способ виртуализации базовой платформы. При этом в общем случае не делается принципиальных различий между “программной” и “аппаратной” реализациями.

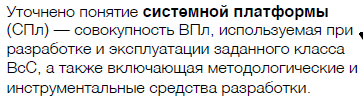
• **Виртуальная вычислительная машина** (ВВМ) – техническое решение, реализующее семантику модели вычислений (англ. MoC).

**Виртуальная машина** - вычислитель, для которого определены правила поведения (например, система команд, условия ввода команд, данных, получения результата, правила синхронизации процесса), позволяющие однозначно описать алгоритм решения задачи. Описание виртуальной машины демонстрирует лишь внешние свойства вычислителя и правила его использования, не касаясь его устройства.

# 11. Понятие платформы (вычислительной, системной, инструментальной…).

Сегодняшняя практика состоит в проектировании на базе готовых вычислительных платформ (аппаратных, программных, инструментальных, конструктивных, …).

источник:презентация пенского architecture-styles-slides из репы на гите



источник:<https://cyberleninka.ru/article/n/arhitekturnye-abstraktsii-v-tehnologii-skvoznogo-proektirovaniya-vstroennyh-vychislitelnyh-sistem>

Вычислительная платформа (ВП). Термин платформно-ориентированное проектирование (platform-based design) был введен совсем недавно специалистами в контексте технологии Codesign [4, 5]. Существуют его различные трактовки. Мы этим термином будем обозначать фиксированную функционально завершенную часть архитектуры ВсС, инвариантную к способу реализации. Специфицируя ВП на уровне технического задания (ТЗ) или используя этот объект на уровне элементной базы, разработчик может в значительной степени формализовать процесс проектирования ВсС. ВП можно рассматривать, как: а) класс элементной базы, наряду с микропроцессорами (МП), интерфейсами, операционными системами реального времени (ОСРВ) и другими объектами; б) способ обеспечения программной и аппаратной совместимости; в) базу для повторного использования разработанных компонент; г) абстрактное ядро ряда совместимых ВсС.

Ист.: Высокоуровневое проектирование встраиваемых систем. Платунов.

Понятие платформы в вычислительной технике на интуитивном уровне последние 10 лет применяется исключительно широко. Например, в рамках одного из ведущих направлений проектирования ВсС и СнК «Platform-Based Design» (PBD) вычислительная платформа определяется как множество проектов, удовлетворяющее некоторому общему условию.

Важнейшее свойство вычислительной платформы – возможность предоставлять заданный уровень абстрагирования от конкретики реализации.

В системе абстракций вычислительная платформа (ВПл) рассматривается как единство «внешнего» и «внутреннего» представления функционально-завершенного и функционально-значимого объекта в составе ВсС.

Платформа в рамках проекта ВсС – техническое решение, фиксируемое в проекте для повторного использования. ВПл выступает основным инструментом повторного использования на архитектурном уровне.

В рамках аспектной модели центральным понятием этапа архитектурного проектирования выступает архитектурная платформа. Это понятие фиксирует инфраструктуру проекта, в которой будет осуществляться генерация, конкретизация, верификация архитектурной модели целевой системы, а также формирование спецификаций для этапа реализации. Таким образом, архитектурная платформа может рассматриваться как объединение следующих элементов проектирования:

* аспектное пространство процесса проектирования (перечень аспектов проектирования);
* модель (модели) вычислений;
* внешние факторы, задающие допустимые соотношения между отдельными аспектами (критерии проектирования);
* перечень зафиксированных шаблонов повторного использования;
* элементная база (в расширенной трактовке).

Фактически, архитектурная платформа выступает в качестве глобального набора решений, ограничений и приоритетов, регламентирующих проектную деятельность как на этапе проектирования, так в значительной степени и на этапе реализации. Она фиксирует концептуальные решения проекта.

Понятие платформы представляется удобным классификационным параметром в данной области проектирования. Можно строить дерево классификаций ВсС на основе понятия проектной платформы (вычислительной или иной платформы, взятой за базу в рамках комплексного или частного проекта ВсС), что позволяет акцентироваться на общих свойствах системы или ее части во всем пространстве свойств, технологий и реализаций ВсС. Пример классификации на базе проектной платформы представлен ниже:

* Вычислительные платформы (процессоры, ОС, MoC и т.д.):
* Платформы промышленных ПК и ПЛК;
* Полуфабрикаты от мультимедиа-индустрии;
* Микроконтроллеры и DSP;
* ПЛИС, ПСнК;
* ASIC, ASSP, ASIP, SOC, NOC;
* «Свободная» кремниевая компиляция.
* Платформы системного ПО (ОС), сетевые, интерфейсные, конструктивные, инструментальные и другие.

# 12. Модель вычислений (MoC).

**Модель вычислений** — математическая модель, которая демонстрирует вычислительные возможности и правила использования субъекта вычислений (актора).

Поведенческий аспект проектирования является моделью вычислений.

(из методы)   
При представлении вычислительной системы как иерархии виртуальных машин каждому уровню такой модели соответствует своя MoC.   
**MoC** можно интуитивно представить, как набор правил, необходимых для построения вычислительного процесса системы. Это парадигма, описывающая протекание вычислительного процесса, способы обмена данными, взаимодействия между отдельными функциональными элементами. Кроме того, MoC предлагает терминологию и примитивы, в базисе которых требуется выражать и описывать целевую систему. MoC описывает природу потоков данных, элементов синхронизации, роль времени в процессе выполнения системой целевой функции. Различные MoC по-разному описывают одни и те же процессы, протекающие в целевой системе. Для больших и сложных систем совершенно нормальное положение дел, когда различные части системы представляются различными MoC.

Примерами MoC являются: модель с дискретными событиями, сеть обработки потоков данных, взаимодействующие конечные автоматы, синхронная модель вычислений, объектно-событийная модель вычислений и денотативно-объектная модель вычислений.

Если рассмотреть целевую систему на некотором уровне абстракции как набор взаимодействующих изолированных блоков (вычислительных компонентов системы), то MoC системы на данном уровне абстракции, используя выразительные средства языка, описывает следующие аспекты системы:

* Поведение вычислительных компонентов;
* Взаимодействие вычислительных компонентов;
* Способы передачи данных и синхронизацию вычислений;
* Способы декомпозиции и агрегации вычислительных компонентов.

MoC должна содержать характеристики системы, важные на данном уровне абстракции. Элементы модели (примитивы, языковые средства, требования и др.), а, следовательно, и вся модель в целом, не должны быть слишком абстрактными или слишком конкретными. Т.е. MoC должна быть в состоянии описать целевую систему.

Кроме собственно описания целевой системы на заданном уровне абстракции, MoC должна обеспечивать разработчику средства работы с этим описанием. Разработчик должен иметь возможность доказывать истинность или ложность определенных утверждений, относительно целевой системы, проверять соответствие определенным требованиям и ограничениям, накладываемым на целевую систему. Инструменты, предоставляемые моделью, должны позволять проводить оценку тех или иных характеристик целевой системы, проводить оптимизацию по выбранным параметрам.

# 13. Программируемые и реконфигурируемые архитектуры.

Занять нишу между заказной, жёстко заданной аппаратурой и процессорными программируемыми реализациями призваны реконфигурируемые системы.

источник:<https://cyberleninka.ru/article/n/rekonfiguriruemye-vstraivaemye-sistemy-i-sistemy-na-kristalle>

В 1959 г. Дж. Эстрин ввел понятие „реконфигурируемая вычислительная машина“ как альтернативу программируемой машине фон Неймана. С этого момента работы в области реконфигурируемых архитектур (РА) ведутся постоянно.

Понятие „реконфигурируемая вычислительная система“ (РВС) меняется от „программируемого кусочка кремния“ (S. Goldstein) до „системы, в которой после ее производства конфигурация подсистем или подсистемы может быть заменена или модифицирована для ее функционирования с определенной целью“ (L. Jozwiak).

В понятие РВС специалисты включают архитектуры от однородных вычислительных сред с различной гранулярностью элементов (например, FPGA) до вычислительных сетей разного масштаба (как пространственно-распределенные конфигурации, так и сети на кристалле). В пределе даже программируемые процессоры общего назначения могут определяться в качестве РВС, так как в них аппаратные блоки организуют различные тракты обработки в рамках выполнения конкретных команд [3].

Сегодня единая классификация РВС отсутствует, их сопоставляют по следующим основным характеристикам: роль и расположение ресурсов в системе; типы реконфигурируемых блоков; уровень гранулярности и степень однородности реконфигурируемых ресурсов; метод, время, степень реконфигурации.

Реконфигурируемость обеспечивает повышение производительности, функциональную гибкость и адаптивность, улучшение комплекса надежностных характеристик, сокращение энергопотребления и размеров. Другие достоинства РА требуют анализа жизненного цикла изделия и возможных подходов к его проектированию: удовлетворение нужд потребителей, быстрая разработка и выведение на рынок, повышение качества продукта, адаптация к новым стандартам, снижение стоимости разработки.

Практическое применение РА сдерживают проблемы архитектурного проектирования и отображения (mapping) прикладных алгоритмов на ресурсы РВС. Они должны решаться в комплексе с высокой степенью автоматизации для режимов статического и динамического реконфигурирования. Использование принципа реконфигурируемости архитектуры позволяет получать вычислительные системы со многими полезными свойствами.

**Определение из методы Платунова:**

Реконфигурируемые вычислительные системы (РВС) — это системы, имеющие возможность менять свою модель вычислений, или, иначе говоря, позволяющие вносить существенные изменения в свою аппаратную часть. РВС занимают нишу между двумя парадигмами создания ВС: чисто программными системами на некотором процессоре с фиксированной аппаратной архитектурой и специализированными аппаратными, реализующими, как правило, потоковую модель вычислений.

Необходимость таких систем обусловлена тем, что классическая вычислительная система на базе архитектуры фон-Неймана имеет множество недостатков. Одна из её проблем – взаимодействие процессора с памятью, так как современная память работает значительно медленнее процессора. Несмотря на наличие механизмов, ускоряющих этот процесс (многоуровневая кэш память, конвейерное исполнение команд, предсказание ветвлений), в большинстве случаев АЛУ процессора простаивает, что снижает эффективность системы в целом.

Эффективность работы процессора немного возрастает при использовании механизма гипертрединга, когда к одному блоку памяти обращается два потока, работающие с одной и той же областью памяти в рамках одного процесса. Но, тем не менее рост производительности, необходимый для высокопроизводительных вычислений, и экономичность, важная во ВсС, плохо достижимы в рамках классических архитектур. Чем больше разрыв между скоростью работы памяти и процессора, тем ниже фактически достигаемая эффективность работы вычислительной системы. Кроме этого, 95% времени выполнения инструкции (см. Рис. 16) приходится на всевозможные вспомогательные и подготовительные действия, и лишь 5% на саму вычислительную операцию – то, ради чего всё, собственно, и организуется.

Стандартным подходом к созданию РВС является реконфигурация при помощи записи значений в некоторые регистры, которые влияют на работу системы, для чего используется электричество (другие способы изменения структуры аппаратуры, например, устанавливаемые «джамперы», к вопросу рассмотрения не относятся). Записанные значения могут храниться после выключения питания, а могут исчезать, в зависимости от реализованной технологии их хранения. Грубо говоря, в предельном случае любую память, используемую в современных компьютерах, можно назвать реконфигурируемой системой.

Аппаратные вычислительные элементы РВС могут быть реконфигурированы однажды, когда система выпускается, раз в какое-то количество лет, для устранения багов, а также для обновлений (статическая реконфигурация), или же раз в несколько часов, чтобы адаптироваться под текущую задачу (динамическая/run-time реконфигурация). Часто реализуется частичная реконфигурация — динамическая реконфигурация некоторой части аппаратуры при неизменной структуре остальной аппаратуры.

# 14. Альтернативы в проектировании ВС: аппаратная/программная реализация.

>> А это разве не то же самое, что в 9 вопросе?

На консультации он сказал: что надо сказать что аппаратная и программная реализация это часть единого целого и надо рассмотреть случаи когда используется только аппаратная или программная реализация и по-рассуждать над тем какой % аппаратной/программной реализации где и когда используется.

# 15. Основные группы требований в техническом задании на специализированную ВС.

На консультации он сказал зачекать госты

источник:https://www.prj-exp.ru/gost/gost\_19-201-78.php

1.4. Техническое задание должно содержать следующие разделы:

* введение;
* основания для разработки;
* назначение разработки;
* требования к программе или программному изделию;
* требования к программной документации;
* технико-экономические показатели;
* стадии и этапы разработки;
* порядок контроля и приемки;
* в техническое задание допускается включать приложения.

------------------------------------

* требования к функциональным характеристикам;
* требования к надежности;
* условия эксплуатации;
* требования к составу и параметрам технических средств;
* требования к информационной и программной совместимости;
* требования к маркировке и упаковке;
* требования к транспортированию и хранению;
* специальные требования.

# 16. Обзор традиционных методик проектирования ВС.

Традиционные процессы разработки: для аппаратуры – синтез, размещение и трассировка, для ПО – компиляция, запуск и отладка

Традиционные методы проектирования (аппаратуры, ПО, систем в целом) представляют собой стадии разработки требований, анализа, проектирования, реализации и верификации.

Ключевыми чертами традиционного процесса проектирования микропроцессорных вычислительных систем следует считать:

1. ручное разбиение системы на аппаратную и программную части (на основе опыта разработчика) на начальном шаге проектирования;
2. раздельное моделирование и последовательное проектирование аппаратуры и программы;
3. ручную интеграцию аппаратной и программной частей проекта;
4. исправление (компенсацию) выявившихся в процессе отладки ошибок за счет изменения программы (с ухудшением характеристик системы);
5. повторное выполнение цикла проектирования при невозможности компенсировать ошибки за счет программной части.

Особенность данного процесса состоит в раннем делении системы на аппаратную и программную составляющие с последующим изолированным их проектированием.

Такой подход приводит к высокой избыточности реализации, выявлению ошибок только в конце - на этапе объединения аппаратуры и программы, устранению ошибок путем практически полного повторного проектирования. Распространенные технологии проектирования ВсС можно разделить на группы по степени влияния применяемой готовой элементной базы на архитектуру создаваемой системы. Можно говорить о следующих группах:

1. на основе готовой вычислительной системы
2. проектирование на основе набора вычислительных модулей (полузаказное);
3. проектирование в условиях свободного выбора элементной базы (заказное).

На практике сегодняшнее массовое проектирование ВсС сводится к полузаказному (модульному) способу или к использованию готовой вычислительной системы.   
Заказное проектирование ВсС применяется в случаях создания систем с “нестандартными” характеристиками и при создании образцов для дальнейшего серийного и массового производства.

Анализ традиционного процесса проектирования ВсС позволяет выделить следующие недостатки:

* неформальное разбиение на аппаратную и программную части на начальном шаге;
* последовательное проектирование аппаратуры и программы;
* раздельное моделирование аппаратуры и программы;
* ручная интеграция аппаратной и программной частей проекта;
* компенсация выявившихся в процессе отладки ошибок за счет изменения программы.

Специалисты отмечают, что использование традиционных методов проектирования в сегодняшних условиях привело к системному кризису в области создания ВсС. Существует растущий разрыв между потенциальными возможностями элементной базы, эффективностью схемного и программного проектирования, требуемыми объемами верификации и тестирования, архитектурным проектированием. Использование распространенных технологий проектирования заставляет разработчиков постоянно повышать уровень "строительных кирпичей", что при отсутствии средств комплексной оптимизации и верификации ведет к эффекту "насыщения" в достижимой сложности и качестве конечного продукта [16].

Проблема состоит в том, что существующий потенциальный объем разработок ВсС не может быть выполнен коллективами, работающими в данной области, в рамках традиционных технологий проектирования либо в требуемые сроки, либо с достаточным качеством. Кризис влияет на качество разработок (массовое появление "сырых" продуктов на рынке, в первую очередь это относится к микросхемам и программному продукту), ограничивается доступная сложность проектируемых ВсС, практически не применяются технологии повторного использования элементов разработок.

# 17. Проектирование на готовой вычислительной платформе.

источник :<https://vk.com/doc81390383_533235091?hash=26407b68fd3e0f8223&dl=abf9b1d0685d67aec0>

Сложившаяся практика проектирования вычислительных систем вообще и ВсС, в частности, как было отмечено выше, состоит в выборе одной из канонических вычислительных платформ (ВП), на которой за счет программной надстройки решается прикладная задача. Задача делится на две части: выбирается база (платформа), база достраивается вверх (за счет программирования в широком смысле) до получения требуемой ВС. Для такого способа проектирования существуют технологические приемы и инструментальные средства. Примерами являются языки программирования, на которых описывается конечная задача, исполнительные устройства (готовые вычислительные машины) и трансляторы. Применяется и второй вариант: выбирается ВП и наряду с достройкой вверх выполняется модификация вниз. В этом случае базовая платформа выступает и в роли прототипа. Такой способ используется реже из-за высокой трудоемкости

# 18. Заказное проектирование.

проектирование в условиях свободного выбора элементной базы (заказное)

Заказное проектирование ВсС применяется в случаях создания систем с "нестандартными" характеристиками и при создании образцов для дальнейшего серийного и массового производства.

# 19. Элементы и технологии повторного использования в проектировании ВС.

Фактически, архитектурная платформа выступает в качестве глобального набора решений, ограничений и приоритетов, регламентирующих проектную деятельность как на этапе проектирования, так в значительной степени и на этапе реализации. Она фиксирует концептуальные решения проекта.

Важнейшей задачей современного этапа развития методик и средств проектирования ВсС является повышение степени повторного использования результатов, получаемых на ранних (высокоуровневых) этапах проектного процесса. В качестве инструмента, обеспечивающего повторное использование концептуальных решений в области ВсС в рамках аспектной модели проектирования, предлагается использовать понятие архитектурной платформы.

(Из второго дока)

В большинстве инженерных разработок процесс проектирования основан на повторном использовании уже имеющихся компонентов.

Повторное использование компонентов позволит существенно сократить расходы на разработку ПО. Только с помощью систематического повторного использования ПО можно уменьшить расходы на его создание и обслуживание, сократить сроки разработки систем и повысить качество программных продуктов.

Метод проектирования ПО, основанный на повторном использовании, предполагает максимальное использование уже имеющихся программных объектов:

1. Повторно используемые приложения.

2. Повторно используемые компоненты.

3. Повторно используемые функции.

Преимущества повторного использования:

* Повышение надежности - компоненты, повторно используемые в других системах значительно надёжнее новых компонентов, так как уже многократно протестированы.
* Эффективное использование специалистов - часть специалистов, выполняющих одинаковую работу в разных проектах, может заниматься разработкой компонентов для их дальнейшего повторного использования, эффективно применяя накопленные знания.
* Ускорение разработки

Для успешного проектирования и разработки ПО с повторным использованием компонентов должны выполнятся следующие условия:

1. Возможность поиска необходимых системных компонентов.
2. При повторном использовании необходимо удостовериться, что поведение компонентов предсказуемо и надежно. В идеале все компоненты должны быть сертифицированы, чтобы подтвердить соответствие определенным стандартам качества.
3. На каждый компонент должна быть соответствующая документация, цель которой — помочь разработчику получить нужную информацию о компоненте и адаптировать его к новому приложению.

Покомпонентная разработка

1. Компонент — это независимо выполняемый программный объект. Исходный код компонента может быть недоступен, поэтому такой компонент не компилируется совместно с другими компонентами системы.
2. Компоненты объявляют свой интерфейс и все взаимодействия с ними осуществляются с его помощью. Интерфейс компонента описывается в терминах параметризованных операций, а внутреннее состояние компонента всегда скрыто.

# 20. Надежность проектирования.

## **Надёжность на этапе проектирования (из вики)**

Надёжность на этапе проектирования является новой дисциплиной и относится к процессу разработки надёжных изделий. Этот процесс включает в себя несколько инструментов и практических рекомендаций и описывает порядок их применения, которыми должна владеть организация для обеспечения высокой надёжности и ремонтопригодности разрабатываемого продукта с целью достижения высоких показателей готовности, снижения затрат и максимального срока службы продукта. Как правило, первым шагом в этом направлении является нормирование показателей надёжности. Надёжность должна быть «спроектирована» в системе. При проектировании системы назначаются требования к надёжности верхнего уровня, затем они разделяются на определённые подсистемы разработчиками, конструкторами и инженерами по надёжности, работающими вместе. Проектирование надёжности начинается с разработки модели. При этом используют структурные схемы надёжности или деревья неисправностей, при помощи которых представляется взаимоотношение между различными частями (компонентами) системы.

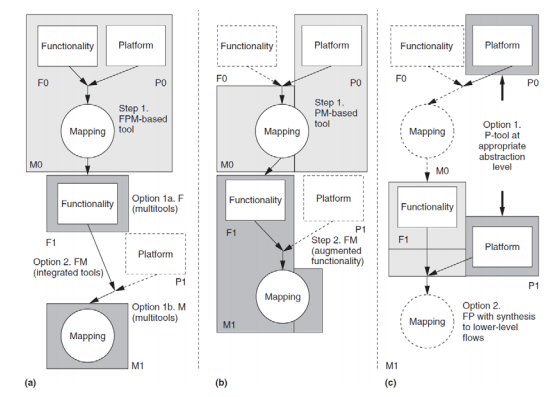
Одной из наиболее важных технологий проектирования является введение избыточности или резервирование. Резервирование — это способ обеспечения надёжности изделия за счёт дополнительных средств и (или) возможностей, избыточных по отношению к минимально необходимым для выполнения требуемых функций (ГОСТ 27.002). Путём введения избыточности совместно с хорошо организованным мониторингом отказов, даже системы с низкой надёжностью по одному каналу могут в целом обладать высоким уровнем надёжности. Однако введение избыточности на высоком уровне в сложной системе (например, на уровне двигателя самолёта) очень сложно и дорого, что ограничивает такое резервирование. На более низком уровне системы резервирование реализуется быстро и просто, например, использование дополнительного соединения болтом.

Существует много методик анализа надёжности, специфических для отдельных отраслей промышленности и приложений. Наиболее общие из них следующие.

* [Анализ видов и последствий отказов](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%B2%D0%B8%D0%B4%D0%BE%D0%B2_%D0%B8_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B8%D0%B9_%D0%BE%D1%82%D0%BA%D0%B0%D0%B7%D0%BE%D0%B2) (АВПО)
* [Имитационное моделирование](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B8%D1%82%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) надёжности
* [Анализ схем функциональной целостности](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D1%86%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8) (СФЦ)
* Анализ опасностей (Hazard analysis)
* Анализ структурных схем надёжности (RBD)
* Анализ деревьев неисправностей
* Ускоренные испытания
* Анализ роста надёжности
* Вейбулл-анализ (анализ эмпирических данных из испытаний и эксплуатации)
* Анализ смеси распределений
* Устранение критичных отказов
* Планирование технического обслуживания, обеспечивающего надёжность ([RCM](https://ru.wikipedia.org/wiki/RCM))
* Анализ диагностики отказов
* Анализ ошибок человека-оператора

Инженерные исследования проводятся для определения оптимального баланса между надёжностью и другими требованиями и ограничениями. Существенную помощь при инженерном анализе надёжности могут оказать программные комплексы для расчёта надёжности.

# 21. Сценарии проектирования ВС: целевая система, платформа, модернизация целевой системы.



Стр 26-28:<https://vk.com/doc81390383_533235091?hash=26407b68fd3e0f8223&dl=abf9b1d0685d67aec0>

(Из второго дока)

Важным в проектировании ВсС является анализ базовых сценариев проектирования, т.к. они во многом определяют постановку задачи или постановка задачи определяет выбор сценария и необходимую методику проектирования. Выделяются три сценария проектирования ВсС:

а) заказной или целевой, когда создается специализированная ВсС «под ключ»;

б) создание вычислительной (проектной) платформы для целевых ВсС;

в) модификация существующих систем или платформ.

Основными являются первые два сценария.

Сценарий 1: новый дизайн приложения из спецификации

Требования этого сценария включают в себя необходимость начать с высокоуровневой спецификации. Эта структура предлагает гибкий ввод спецификаций - без привязки к конкретному стилю реализации или платформе; сервисы, которые помогают перемещать абстрактный дизайн в сторону более ограниченной версии (например, алгоритмы, которые могут реализовывать функциональность); и независимое отображение функциональности на алгоритмические структуры, которые позволяют повторно использовать функциональную спецификацию.

Сценарий 2: разработка новой интеграционной платформы

Этот сценарий описывает разработку новой интеграционной платформы: аппаратную архитектуру, архитектуру встроенного программного обеспечения, методологии проектирования, рекомендации по проектированию и стандарты моделировани. В отличие от первого сценария, этот не связан с разработкой конкретного приложения, а скорее с разработкой субстрата для реализации нескольких приложений.

Сценарий 3: Интеграция устаревшего дизайна

В этом случае сложно разделить функциональность и архитектуру, поскольку в большинстве встроенных систем документация относится к конечной реализации, а не к ее исходным спецификациям. Если модификация дизайна необходима для реализации дополнительных функций, очень трудно определить, как новая функциональность повлияет на существующий дизайн.

# 22. Встроенное программное обеспечение.

Стр 15

<https://vk.com/doc81390383_533235091?hash=26407b68fd3e0f8223&dl=abf9b1d0685d67aec0>

(Из второго дока)

ВПО – это один из вариантов реализации функциональности ВсС, которая с тем же успехом может быть реализована как аппаратный компонент. Попытки прямого переноса методов и технологий, традиционных для индустрии ПО, в область ВПО привели сегодня к кризису в этой области.

Рассмотрим более подробно проблемы создания встроенного программного обеспечения.

* необходимость увеличения степени повторного использования;
* создание средств моделирования нефункциональных свойств;
* усиление роли в проектировании уровня архитектуры систем и ПО;
* проверка достоверности и верификация;
* адаптация ПО и аппаратуры посредством использования реконфигурируемых архитектур.

Вторым («обобщающим») вариантом трактовки ВПО может выступать определение ВПО как всей совокупности алгоритмического обеспечения ВсС. Этот подход хорошо укладывается в модель единого сквозного цикла проектирования ВсС, в которой важнейшее место занимают этапы высокоуровневого проектирования, широко используются абстракции организации вычислительного процесса.

В состав ВПО может включаться:

1. только то ПО, которое «работает» в режиме run-time внутри ВсС на последовательных интерпретаторах;
2. то, что на этапе проектирования технологически представлялось, как программирование и было в дальнейшем реализовано в ВсС в любом виде (в виде аппаратных блоков, конфигурационного обеспечения, традиционного программного кода);
3. вся совокупность деятельности по созданию алгоритмического наполнения ВсС, которая присутствует на этапах design-time и run-time.

# 23. Надежность программного обеспечения ВС.

Надежность работы вычислительной аппаратуры следует рассматривать совместно с программным обеспечением как надежность вычислительного процесса. Под надежностью программного обеспечения (ПО) будем понимать свойство программы выполнять заданные функции, сохранять свои характеристики в установленных пределах при определенных условиях эксплуатации. Надежность ПО определяется его безотказностью и восстанавливаемостью.

определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью.

Степень надежности характеризуется вероятностью работы программного продукта без отказа в течение определенного периода времени.

Надежность программного обеспечения есть вероятность его работы без отказов в течение определенного периода времени, рассчитанная с учетом стоимости для пользователя каждого отказа. Слово «вероятность» в определении по существу означает вероятность того, что пользователь не введет в систему неко­торый конкретный набор данных, выводящий систему из строя. Надежность также не является внутренним свойством прог­раммы. Она во многом связана с тем, как программа исполь­зуется.

Надежность программного обеспечения существенно отли­чается от надежности аппаратуры. Программы не изнашиваются, поломка программы невозможна. Таким образом, надежность программного обеспечения — есть следствие исключения ошибок проектирования, т.е. ошибок, внесенных в процессе разработки программного обеспечения.

Надежность является составной частью более общего понятия — качества.  
Качественная программа, например, не только надежна, но и компактна, совместима с другими программами, эффективна, удобна в сопровождении, вполне понятна. Можно добавить: программа должна быть разработана в срок и в пределах бюджетной стоимости.

Среди прочих характеристик качества программ надежность стоит на первом месте, и поэтому дальнейшие вопросы раз­работки программного обеспечения рассматриваются через приз­му надежности.

Зависимости стоимости и вероятности обнаружения и исправления ошибок от времени проектирования программного обеспечения

# 24. Задачи инструментария ВС, классификация инструментальных средств.

**Инструмента́льное програ́ммное обеспе́чение** — [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), предназначенное для использования в ходе [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), [разработки](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) и сопровождения программ.К этой категории относятся программы, предназначенные для разработки программного обеспечения:

* [ассемблеры](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80) — компьютерные программы, осуществляющие преобразование программы в форме исходного текста на [языке ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0) в [машинные команды](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) в виде [объектного кода](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4).
* [трансляторы](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) — программы или технические средства, выполняющие трансляцию программы.
* [компоновщики](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%BE%D0%B2%D1%89%D0%B8%D0%BA) (редакторы связей) — программы, которые производят компоновку — принимают на вход один или несколько объектных модулей и собирают по ним исполняемый модуль.
* [препроцессоры](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) исходных текстов — это компьютерные программы, принимающие данные на входе и выдающие данные, предназначенные для входа другой программы, например, такой, как компилятор
* [отладчики](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA) (debugger) являются модулями среды разработки или отдельными приложениями, предназначенными для поиска ошибок в программе.
* [текстовые редакторы](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80) — компьютерные программы, предназначенные для создания и изменения текстовых файлов, а также их просмотра на экране, вывода на печать, поиска фрагментов текста и т. п.
  + специализированные [редакторы исходных текстов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D1%85_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2) — текстовые редакторы для создания и редактирования исходного кода программ. Специализированный редактор исходных текстов может быть отдельным приложением, или быть встроен в интегрированную среду разработки (IDE).
* [библиотеки подпрограмм](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) — сборники подпрограмм или объектов, используемых для разработки программного обеспечения.
* [редакторы графического интерфейса](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0&action=edit&redlink=1).
* [Средства автоматизации разработки программ](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC) (CASE-средства)

# 25. Инструментальные резидентные и кросс-средства.

Инструментальные средства — см. выше в вопросе 24.

**Инстр.средства** — комплекс средств, предназначенный для разработки и отладки программного обеспечения. В него обычно входят трансляторы, интерпретаторы, различного рода отладчики и другие программные средства.

Инструментальное ПО — категория, отличная от системного ПО (ОС, драйверы) и прикладного ПО.

Для проведения этапа трансляции и отладки разработанных программ используются различные инструментальные средства (редакторы, трансляторы, компоновщики, отладчики), которые делятся на резидентные и кросс-средства.

* **Кросс-средства** – это программы, которые выполняются не на микроЭВМ, для которой разрабатываются программы, а на какой-либо другой машине. Пример: кросс-компилятор, который запускается на персональном ПК с архитектурой x86-64, но переводит код на языке Си в машинный код для процессоров архитектуры ARM (например, STM32).
* **Резидентные средства** – это те инструментальные программы, которые выполняются на тех же микропроцессорных системах, для которых и разрабатываются программы. Пример: обычный транслятор, запускаемый на той же машине, на которой будет выполняться транслируемая программа.