Вариант 1.

1. Отличие СнК от ИС. Примеры

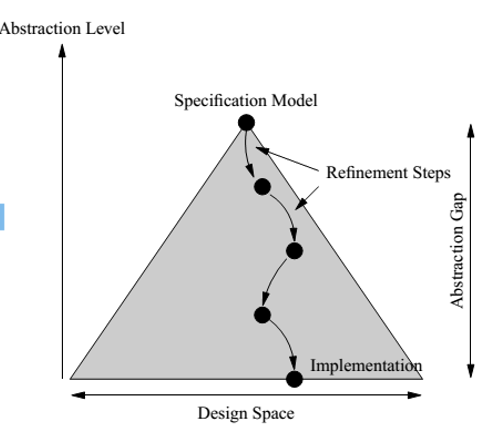
ИС – разные степени интеграции (кол-во элементов на кристалле)  
СнК – готовый функционал.

ИС – как кирпич

1. Основные хар-ки и особенности системного подхода к проектированию system-based design

Процесс проектирования начинается с создания формальной спецификации системы, исходя из требований ТЗ

На каждом шаге происходит постепенная детализация спецификации системы вплоть до выпуска готовой СнК



1. Модель вычислений. Два примера

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.

1. СенК. Отличие от СнК

С развитием уровня техники у производителей процессорных ядер возникла проблема, не позволяющая дальше увеличивать тактовую частоту работы. Для увеличения производительности процессоров было придумано встраивать сразу несколько ядер в один процессор и распараллеливать задачи. Однако с увеличением числа ядер стало видно, что после 8 ядер производительность практически не растет. Это связано с способом общения между процессорными ядрами посредствам шины. Когда много ядер пытаются одновременно обратиться к шине возникают взаимные блокировки и простои в работе.

Network on Chip, или сеть с коммутацией пакетов на одном кристалле – это решение, призванное заменить стандартную шину в многоядерных процессорах и решить проблему увеличения числа ядер. В данной архитектуре каждое ядро непосредственно соединено только с маршрутизатором. Маршрутизаторы в свою очередь объединены в одну сеть на подобии обычной компьютерной сети. Маршрутизаторы выбирают путь прохождения сигнала к другому маршрутизатору, т.е. к другому ядру, никак не мешая общению других ядер между собой. Такой подход значительно упрощает топологию схем решает все проблемы масштабируемости.

1. Fabless-компания. 3 примера (название – сфера деятельности)

**беспроизводственная компания**[[*источник не указан 430 дней*](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8)] — модель организации бизнеса в электронной промышленности, при которой [компания](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D1%80%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)-производитель специализируется только на разработке и продаже [микроэлектроники](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0), но не имеет собственных производственных мощностей, то есть пользуется услугами [OEM](https://ru.wikipedia.org/wiki/OEM)-предприятий для производства. То есть, чтобы изготовить продукцию, спроектированную в собственных лабораториях[[1]](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%81%D1%84%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F#cite_note-1), беспроизводственная компания передаёт технологию и размещает заказ на специализированном производстве других компаний, которые часто называют кремниевыми заводами.

Xilinx – производство fpga

Nvidia – GPU,

Broadcom – интегральные схемы

1. Технология высокоуровневого синтеза. Почему высокоуровневый.

Задача систем высокоуровневого синтеза (High-Level Synthesis — HLS) заключается в отображении поведенческой модели алгоритма, описанной на языке высокого уровня, на специализированную аппаратную архитектуру.

Всё дело в том, что HLS пакеты, помимо традиционного RTL кода написанного на SystemC умеют синтезировать и чисто поведенческий (“untimed”) код, автоматически вставляя регистры, там где это необходимо

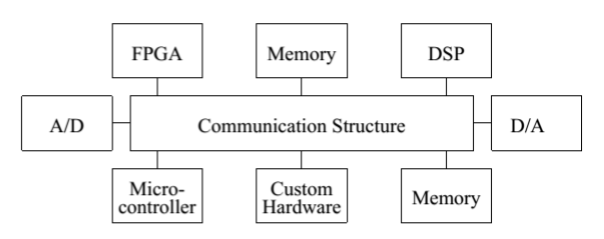
1. +- аппаратной реализации алгоритмов обработки данных в СнК. Примеры алгоритмов и ситуаций, когда полезно, когда нет.

*Аппаратная реализация* подразумевает использование разнообразных функциональных блоков: регистров, сумматоров, шифраторов и дешифраторов, счётчиков, линий задержек, устройств памяти, умножителей, сдвигателей, логических элементов, интегральных и больших интегральных схем, программируемых логических матриц и т. п. Совокупность функциональных блоков и связей между ними определяет реализуемый алгоритм. Достоинство аппаратной реализации состоит в очень высоком быстродействии, что позволяет обрабатывать сигналы при частоте дискретизации в десятки МГц. Это достигается применением функциональных блоков на базе ТТ-логики, распараллеливанием операций и узкой направленностью (специализацией) создаваемых устройств (например, для реализации алгоритма БПФ в радиолокационных системах).  
  
С другой стороны, аппаратная реализация, ориентированная на решение узкоспециальных задач, подразумевает создание систем с жёсткой логикой, когда любое изменение алгоритма требует изменения структуры устройства, т. е. введения дополнительных функциональных блоков, что является недостатком. Кроме того, аппаратная реализация приводит к большому потреблению энергии и к необходимости организовать теплоотвод. Всё это вместе определяет высокую стоимость аппаратной реализации, причём проектирование, изготовление и отладка оказываются весьма трудоёмкими при больших временных затратах.

Вариант 2.

1. СнК. Структурная схема. Составляющие

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

  
С другой стороны, технология проектирования вычислительных систем, выполненных в виде интегральной схемы, ключевыми элементами которой являются

Методы и средства системного проектирования, ориентированные на создание гетерогенной (аппаратно-программной) вычислительной платформы.

Повторное использование в проекте готовых библиотечных «аппаратных» блоков (IP-компонент).

Следование определенному набору стандартов, регламентирующих правила упаковки и вывода на рынок, как всей системы, так и отдельных её компонент (IP-компонент).

1. IP-ядро. Отличие жестких и мягких, примеры.

IP-ядро (Intellectual Property) – интеллектуальная собственность (продукт).

Русский термин – сложно-функциональный блок (СФ-блок).

**Soft IP cores** are IP blocks generally offered as synthesizable RTL models. These are developed in one of the Hardware description language like SystemVerilog or VHDL.

Sometimes IP cores are also synthesized and provided as generic gate level netlist which can be then mapped to any process technologies. This also falls under Soft IP cores. The advantage of Soft IP cores is that those can be customized in the back end Placement and Routing flow by a consumer to map to any process technologies.

**Hard IP cores** on the other hand are offered as layout designs in a layout format like GDS which is mapped to a process technology and can be directly dropped by a consumer to the final layout of the chip. These cores cannot be customized for different process technologies.

Generally **digital logic cores** are developed and licensed as **Soft IP cores**. eg: a DRAM controller IP, Ethernet MAC IP, AMBA bus procotol IPs etc

**Analog and Mixed signal** logic designs for serdes, PLLs, ADC or DAC, Phy layer logic for DDR, PCIE etc are generally developed and licensed as **Hard IP cores.**

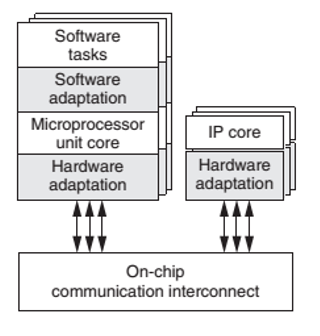
IP-библиотеки базового уровня • IP-модули памяти • Интерфейсные IP-модули • Процессорные IP-ядра

1. Component-based design

Прикладная задача решается с помощью готовых IP-блоков

Акцент смещается с проектирования вычислительных узлов на проектирование инфраструктуры их взаимодействия

Основные проблемы: выбор топологии, арбитраж доступа к среде передачи, диспетчеризация процессов.



1. Отличие ASIC, StructuredASIC, FPGA

программируемая пользователем вентильная матрица **field-programmable gate array** (FPGA) и специально разработанная специализированная интегральная схема Application-specific integrated circuit (ASIC).

FPGA состоит из внутренних аппаратных блоков с программируемыми пользователем межсоединениями для настройки операций для конкретного приложения. В отличие от других упомянутых устройств, соединения между блоками могут быть легко перепрограммированы, изменяя внутреннюю работу аппаратного обеспечения и позволяя FPGA приспосабливаться к изменениям проекта или даже поддерживать новое приложение в течение срока службы устройства. Такая гибкость делает FPGA отличным выбором для приложений, таких как цифровое телевидение, бытовая электроника, системы кибербезопасности и беспроводная связь.

На другом конце спектра имеются специализированные микросхемы ASIC, разработанные специально для конкретного применения. Они имеют только блоки, необходимые для оптимальной работы приложения, включая процессор, графический процессор, память и так далее. Хотя разработчики могут использовать сторонние IP-ядра, такие как процессор ARM Cortex, или предварительно разработанные блоки для стандартных функций, таких как физический уровень Ethernet, ASIC является разработкой с нуля. Лучше всего подходит для приложений с большим объемом вычислений. Например, Tensor Processing Unit (TPU) – это ускоритель, разработанный Google специально для машинного обучения нейронных сетей. Google также предлагает доступ к TPU сторонним компаниям через облачные вычисления.

**Structured ASIC** is an intermediate technology between [ASIC](https://en.wikipedia.org/wiki/Application-specific_integrated_circuit) and [FPGA](https://en.wikipedia.org/wiki/FPGA), offering high performance, a characteristic of ASIC, and low [NRE](https://en.wikipedia.org/wiki/Non-recurring_engineering) cost, a characteristic of FPGA. Using Structured ASIC allows products to be introduced quickly to market, to have lower cost and to be designed with ease.

In a FPGA, interconnects and logic blocks are programmable after fabrication, offering high flexibility of design and ease of debugging in prototyping. However, the capability of FPGAs to implement large circuits is limited, in both size and speed, due to complexity in programmable routing, and significant space occupied by programming elements, e.g. SRAMs, MUXes. On the other hand, ASIC design flow is expensive. Every different design needs a complete different set of masks. The Structured ASIC is a solution between these two. It has basically the same structure as a FPGA, but being mask-programmable instead of field-programmable, by configuring one or several via layers between metal layers. Every SRAM configuration bit can be replaced by a choice of putting a via or not between metal contacts.

A number of commercial vendors have introduced structured ASIC products. They have a wide range of configurability, from a single via layer to 6 metal and 6 via layers. Altera's Hardcopy-II, eASIC's Nextreme are examples of commercial structured ASICs

1. САПР в области СнК. 3 примера, для чего

Vivado HLS -САПР Xilinx, предназначенная для создания цифровых устройств с применением языков высокого уровня.

Catapult HLS, Stratus HLS

Создают RTL дизайн на основе с/с++, системС кода

1. Формализация результатов проектирования. Роль в процессе проектирования

Формализация – это процесс, результатом которого является представление информации об объекте в форме, доступной для хранения, передачи и обработки другими объектами (человеком, техническими средствами).

Цели формализации – автоматизация процесса проектирования и увеличение доли компонентов повторного использования.

1. +- синтеза аппаратных блоков из реализация на С/С++

Может быть использован для разработки как ПО, так и аппаратного обеспечения

Имеется связь с аппаратурой

Накоплена большая кодовая база в области цифровой обработки сигналов

Понятен для специалистов разных предметных областей

Конструкции С/С++, которые нельзя синтезировать в современных САПР

Системные вызовы Динамическое выделение памяти Приведение указателей к типу void \* Рекурсивные функции