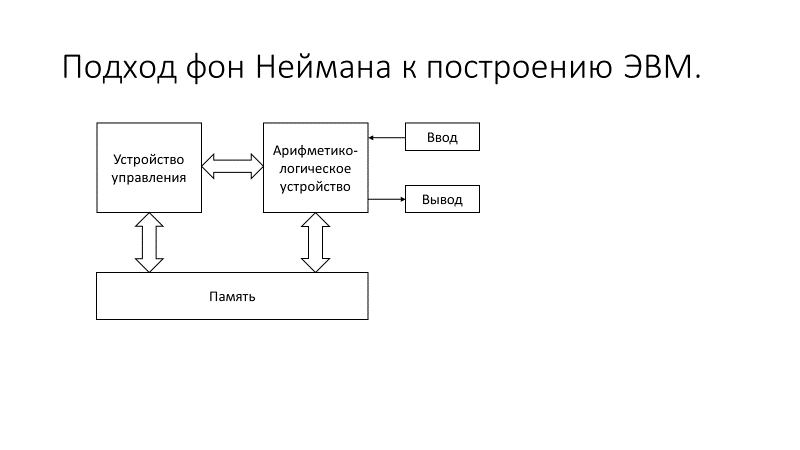
|  |  |
| --- | --- |
| ДИСЦИЛИНА | **Схемотехника устройств компьютерных систем Часть 2** |
|  |  |
| ИНСТИТУТ | **ИТ** |
| КАФЕДРА | **вычислительной техники** |
|  |  |
| ВИД УЧЕБНОГО | **Лекция** |
| МАТЕРИАЛА |  |
| ПРЕПОДАВАТЕЛЬ | **Тарасов И.Е.** |
|  |  |
| СЕМЕСТР | 3 |
|  |  |

1. Основные понятия курса

1.1 Состав компьютерной системы

На рис. 1.1 показана архитектура электронно-вычислительной машины (ЭВМ), предложенной фон Нейманом, который заложил теоретические основы вычислительной техники в середине 20 века. В этой схеме данные, которые обрабатывает ЭВМ, находятся в памяти. Операции по преобразованию данных выполняются арифметико-логическим устройством (АЛУ) на основе сигналов, формируемых устройством управления. В ЭВМ имеются также устройства ввода и вывода данных.



*Рисунок 1.1. Архитектура ЭВМ по фон Нейману*

В схеме фон Неймана не накладываются ограничения на реализацию, форматы данных или типы команд. Устройства ввода и вывода также могут быть различными. Сутью предложенного подхода является описание взаимодействия между компонентами ЭВМ, параметры которых могут уточняться и развиваться по мере перехода к новым технологиям и появлению новых потребностей.

Современной тенденцией является повсеместное использование вычислительных устройств, в том числе для решения задач, которые исходно не требовали вычислителей. Например, устройства «умного дома», носимой электроники, распределенные измерительные устройства, бытовая техника реализуются на основе компьютерной системы. Такие компьютерные устройства образуют широкий класс *встраиваемых* (embedded) систем. Требования к производительности вычислений, объему памяти, типам устройств ввода-вывода и другим параметрам в этом случае различаются очень существенно. Еще больше они различаются при рассмотрении крупных категорий вычислительных систем, которые можно условно представить следующим списком:

- встраиваемые системы;

- мобильные компьютеры (смартфоны, планшеты, ноутбуки);

- стационарные компьютеры (десктопы, рабочие станции);

- серверы, рабочие станции центров обработки данных (ЦОД);

- суперкомпьютеры.

По приведенному списку можно сделать наблюдение о том, что указать какой-то универсальный подход к проектированию невозможно. В случае суперкомпьютера важно обеспечить высокую производительность вычислений и большой объем памяти. Для смартфона, работающего от аккумулятора, важно также время работы, поэтому обеспечивать чрезмерную производительность ценой повышенного энергопотребления будет неправильным. Поэтому по мере развития потребностей в вычислительной технике остается актуальным вопрос поиска новых архитектур, разработки новых процессоров, устройств ввода-вывода и систем на их базе.

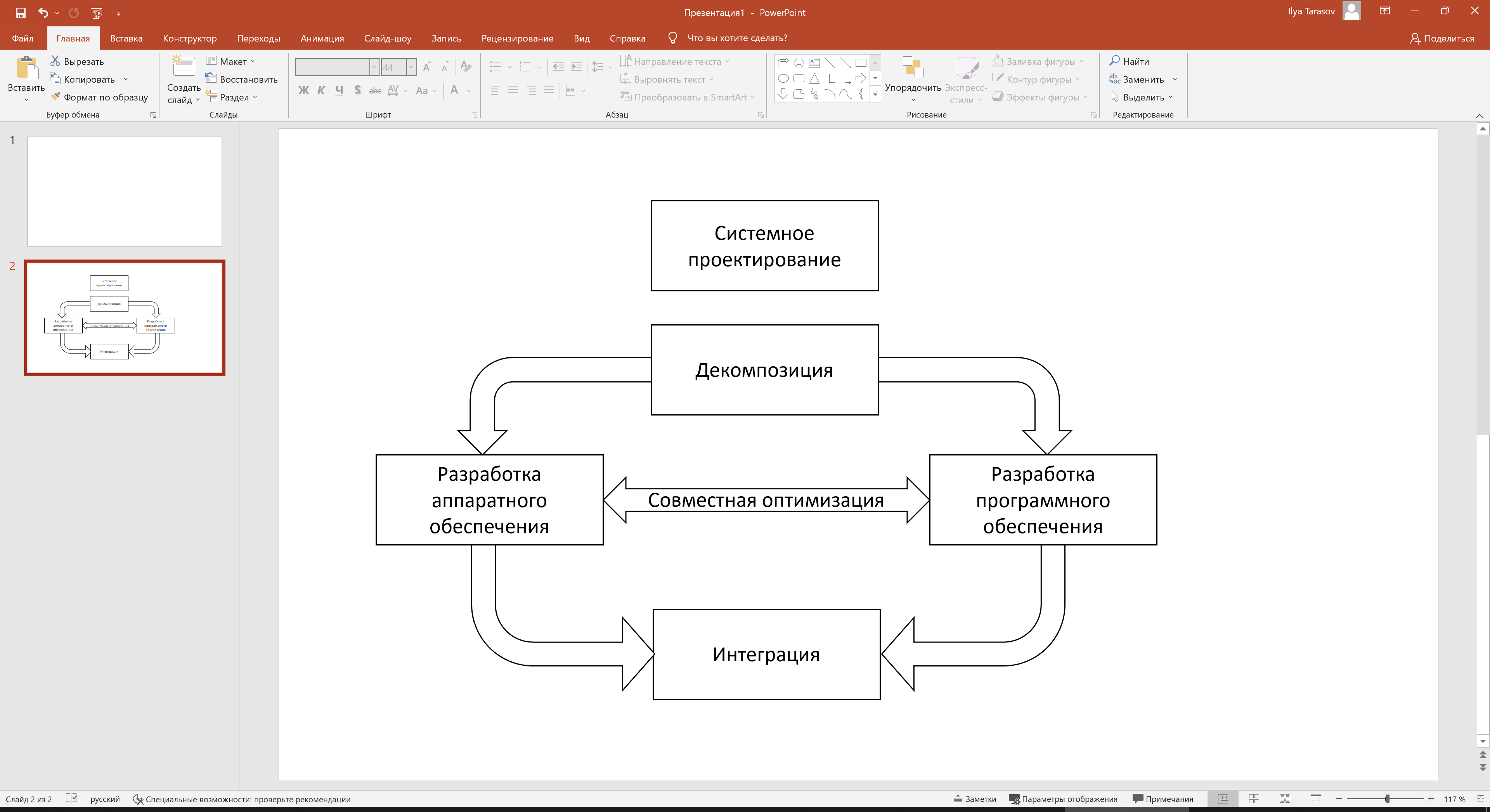
1.2 Маршрут проектирования компьютерной системы

При проектировании компьютерной системы нужно учитывать, что разработанное аппаратное обеспечение будет взаимодействовать с программным обеспечением. Может сложиться ситуация, когда операции, требуемые программе, не смогут быть выполнены вообще или будут выполняться длительное время. Это ухудшит характеристики системы в целом, вплоть до невозможности ее использования. Напротив, если часто используемые операции будут выполняться быстро, эффективность аппаратного обеспечения окажется существенно выше. На этом пути всегда существуют компромиссы – например, полностью ориентировать процессор на выполнение одной определенной программы часто нерационально.

Для современной ситуации в области вычислительной техники характерна тенденция к разработке частично специализированных вычислительных систем. Они обозначаются термином «проблемно-ориентированные системы» (domain-specific), который отражает тот факт, что они нацелены на преимущественное решение одной задачи или подкласса похожих задач. Например, графический сопроцессор (GPU) хорошо выполняет преобразования координат в трехмерном пространстве и вычисляет положение текстур на экране с учетом масштабирования, однако в целом не может запускать программы, предназначенные для центрального процессора. С другой стороны, центральный процессор при построении изображений на экране покажет слишком низкую производительность.

Подход к проектированию аппаратной части с учетом требования будущего программного обеспечения обозначается термином «совместное проектирование» (в зарубежной литературе «hardware & software co-design»). Это не относится к какому-то одному общепризнанному алгоритму или программному продукту. Процесс проектирования в этом случае заключается в разработке варианта аппаратного обеспечения и его поэтапной модификации по мере проверки эффективности программ, которые предполагается на нем запускать. Для такой проверки изготовление микросхем не требуется, вместо этого используется их компьютерное моделирование.

Основные этапы проектирования компьютерной системы проиллюстрированы на рис. 1.2.



*Рисунок 1.2. Основные этапы проектирования компьютерной системы*

Проектирование системы начинается с этапа системного проектирования. Результаты этого этапа будут использоваться для *декомпозиции* – распределения задач между аппаратным и программным обеспечением. Это распределение не обязано быть окончательным, поскольку в процессе разработки могут выявиться дополнительные сведения, которые заставят перенести задачу из одной подсистемы в другую.

Например, программист может выяснить, что определенная функция занимает слишком много времени, снижая общую производительность системы. Общим подходом с точки зрения системного анализа является перенос этой функции в аппаратную подсистему, для чего нужно разработать соответствующее устройство, которое обеспечит выполнение этих действий на уровне аппаратуры без участия процессора.

Не существует единственного общепризнанного подхода к организации подобных проектов. Например, после выхода нового процессора основной идеей проекта может быть обеспечение этого процессора соответствующими программами. В этом случае сам процессор не будет подвергаться модификациям, однако для него можно разработать устройства ввода-вывода и оптимизировать существующее программное обеспечение.

Если проект фокусируется на определенном программном продукте, возможна разработка аппаратных компонентов, которые будут ускорять его выполнение (или как-то еще способствовать улучшению условий использования – например, путем уменьшения потребляемой мощности). Примером могут быть системы беспроводной связи или нейропроцессоры. В этих случаях уже разработанные алгоритмы получают аппаратное обеспечение, которое ускоряет выполнение соответствующих программ, разгружая центральный процессор.

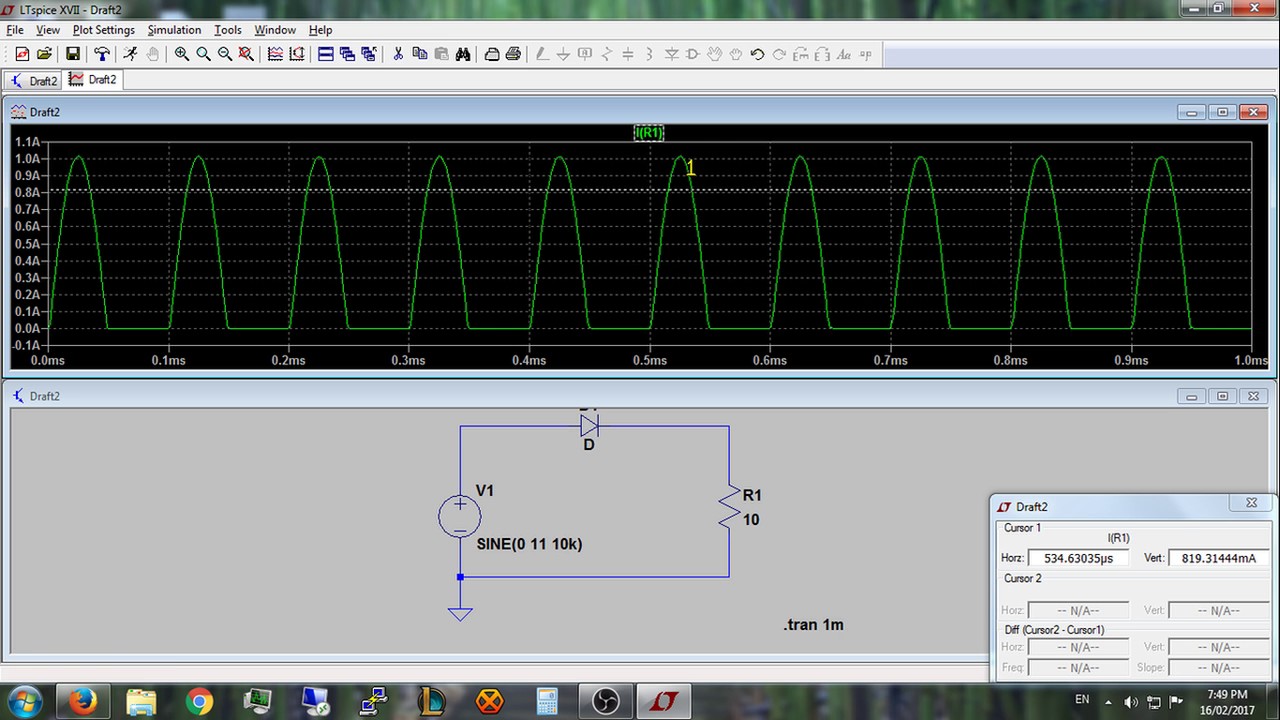
Организацию проектирования компьютерных систем следует выполнять с привлечением методов работы из области управления проектами.

1.3 Цифровая, измерительная и силовая электроника в составе компьютерной системы, методы и инструменты их проектирования.

Электронные компоненты, применяемые в компьютерной системе, можно разделить на цифровые и аналоговые. Цифровые компоненты оперируют дискретными состояниями сигналов (0 и 1), тогда как аналоговые используют непрерывный диапазон напряжений и токов. В свою очередь, аналоговые компоненты можно разделить на измерительные (с приоритетом на обеспечение высокой точности измерений) и силовые (оперирующие сигналами высокой мощности – например, регулирующими ток, протекающий через нагреватель или обмотку электромотора).

Аналоговые компоненты требуют применения других средств проектирования и ставят более сложные задачи. Это связано с тем, что в цифровых схемах для обеспечения уровня 0 или 1 достаточно, чтобы напряжение находилось в определенном диапазоне значений, а для аналоговой техники изменения уровня сигнала недопустимы, поскольку невозможно понять, что это такое – результат действия помех или действительные значения напряжений и токов. Параметры электронных схем зависят от температуры и подвержены технологическому разбросу. Для цифровых схем это может приводить к снижению допустимой тактовой частоты, но для аналоговых может привести к существенным отклонениям от ожидаемого поведения.

На рис. 1.3 показан графический интерфейс системы моделирования аналоговых схем LTSpice. Аббревиатура SPICE означает «Simulation Program with Integrated Circuit Emphasis» и относится к широко распространенному подходу к описанию и моделированию электронных схем. Начальная реализация подобной программы относится к началу 1970-х годов и на протяжении развития электронной техники появлялись многочисленные варианты программного обеспечения и несколько модификаций стандарта.



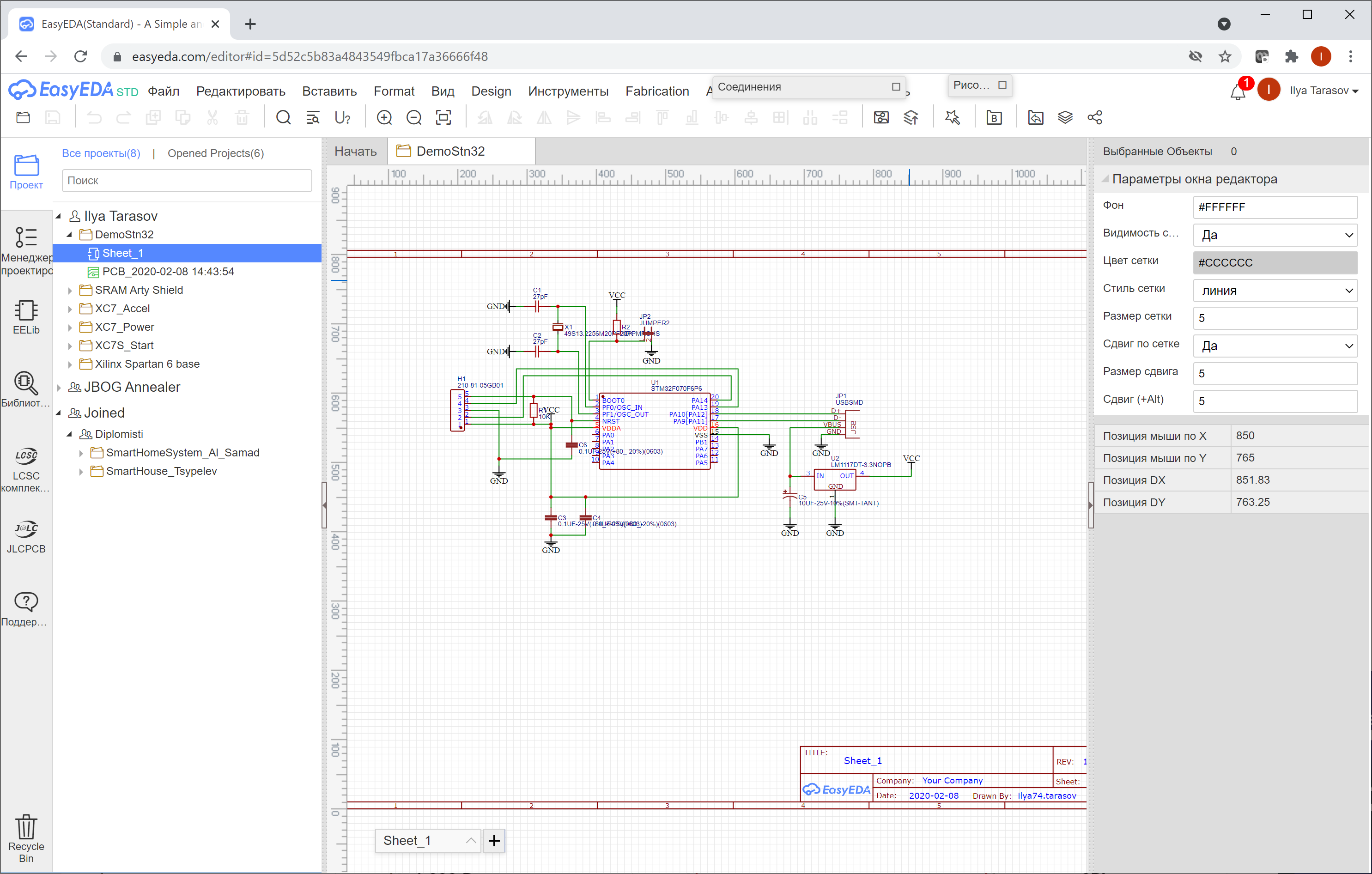
*Рисунок 1.3. Интерфейс системы моделирования аналоговых схем LTSpice*

Общей тенденцией вычислительной техники является перенос основных операций по обработке данных в цифровую часть системы. Например, вместо установки усилителя аналоговых сигналов, умножающего напряжение на какую-то величину, применяется аналого-цифровое преобразование, в результате которого аналоговый сигнал преобразуется в пропорциональный цифровой код, который может быть уже умножен на нужное число соответствующей операцией процессора. Такой подход уменьшает влияние помех на результат, поскольку чем меньше аналоговых цепей находится в составе устройства, тем меньше оказывается мест, на которые могут подействовать помехи.

При этом измерительная и силовая электроника относятся к таким элементам, которые невозможно заменить цифровыми устройствами. Измерительные схемы обычно основаны на микросхемах АЦП (аналого-цифровой преобразователь), однако эти микросхемы работают в определенном диапазоне входных аналоговых сигналов. Поэтому, если измеряемые сигналы не соответствуют диапазонам работы используемых АЦП, требуется применение дополнительных компонентов, усиливающих (или ослабляющих) аналоговые величины по мере необходимости.

Применение силовых компонентов обусловлено тем, что цифровые схемы не могут обеспечить большую выходную мощность. Уровень выходного напряжения составляет 5 или 3,3 В, а выходной ток измеряется десятками миллиампер. Этого достаточно для сигнального светодиода, но такие элементы, как устройство освещения, электромотор или нагреватель, не могут питаться от выхода цифровой микросхемы. Для решения этой задачи в составе системы присутствуют мощные переключающие элементы (обычно транзисторы), которые способны оперировать высокими уровнями напряжения и переключать токи в единицы или десятки ампер. Управление такими элементами осуществляется цифровыми сигналами.

На рис. 1.4 показан внешний вид системы автоматизированного проектирования EasyEDA. Это пример САПР для разработки принципиальных электрических схем и печатных плат, которая позволяет использовать как цифровые, так и аналоговые компоненты.



*Рисунок 1.4. Внешний вид онлайн-САПР EasyEDA в режиме разработки принципиальной электрической схемы*

При проектировании компьютерной системы необязательно использовать полный цикл проектирования аналоговых компонентов и их моделирование в системе SPICE. Измерительные устройства, силовые подсистемы, устройства связи и другие подобные элементы могут быть применены в качестве готовых модулей, производимых сторонними организациями.

1.4 Проектирование цифровых устройств

При проектировании цифровых устройств имеется несколько этапов, на каждом из которых проводятся различные виды работ. На рис. 1.5 схематично показаны основные этапы проектирования и характерные для них проблемы.

*Рисунок 1.5. Основные этапы проектирования цифровых устройств в составе компьютерной системы*

На уровне системного моделирования используются высокоуровневые средства проектирования, включая программы на обычных языках программирования высокого уровня, имитирующие выполнение действий компонентами компьютера. Данный этап нужен для того, чтобы с минимальной трудоемкостью проверить работу будущей системы в основных сценариях ее использования, не вдаваясь в технические детали. Может быть написана программа-эмулятор.

Согласно ГОСТ 15971-90, эмуляция определяется следующим образом: «имитация функционирования одного устройства посредством другого устройства или устройств вычислительной машины, при которой имитирующее устройство воспринимает те же данные, выполняет ту же программу и достигает того же результата, что и имитируемое».

Таким образом, программа-эмулятор может выполнять операции над числами в массиве данных, предполагая, что в будущем этот массив будет заменен на микросхему памяти. Аналогично, выполняя операции над переменными, можно считать, что таким образом проверяется работа регистров будущего процессора.

На уровне, обозначенном как RTL (Register Transfer Level), производится разработка принципиальной электрической схемы. На современном уровне проектирование путем рисования графических символов и соединения их линиями используется ограниченно из-за очень большого объема такой схемы. Этот способ проектирования может быть использован для отдельных элементов, или наоборот, для представления взаимодействия между крупными компонентами системы на верхнем уровне (в этом случае схема становится не «принципиальной электрической», а «структурной» или «схемой соединений», в зависимости от изображаемых на ней элементов).

Основным инструментом описания цифровых схем являются языки описания аппаратуры (HDL, Hardware Description Language). На этом уровне необходимо решить ряд задач, которые были неактуальны для более высокого уровня проектирования. Например, выполнить ряд требований по обеспечению их стабильной работы с учетом применяемой технологии. Они будут рассмотрены далее в этом пособии.

На топологическом уровне выполняется разработка полупроводникового кристалла. Если применяется готовая элементная база или программируемая логическая интегральная схема (ПЛИС), ее изготовление не требуется, однако для ПЛИС проводится размещение и трассировка компонентов. Этот этап является достаточно длительным, поэтому при переходе к нему уже нецелесообразно экспериментировать со схемой или изменять алгоритмы – это приведет к повторному проведению всех операций по размещению и трассировке.

Разработчик не обязан одинаково хорошо владеть всеми видами проектирования. В составе коллектива могут быть взаимодействующие специалисты, каждый из которых выполняет работы в соответствии со своими навыками, однако представляет и проблемы коллег, чтобы не принимать технические решения, которые затруднят их работу. Как и для совместной разработки программного и аппаратного обеспечения, между показанными уровнями возможно взаимодействие для совместной оптимизации компонентов проекта на разных этапах его выполнения.

1.5 Уровни проектирования

Уровни проектирования позволяют сосредотачиваться на отдельных важных аспектах разработки, не выполняя постоянно полный комплекс работ по проектированию. Например, если необходимо определить требуемый объем памяти для процессора, для этого можно не разрабатывать процессор целиком, включая его корпус, печатную плату для установки и микросхему памяти. Моделирование выполнения программы вполне способно выяснить, сколько памяти достаточно для ее выполнения. Можно привести много подобных примеров, когда целесообразность изменения схемы выясняется без ее полного проектирования и тем более изготовления. Поэтому проектирование разбивается на крупные уровни, которые позволяют привлекать для работы специалистов разной направленности.

На уровне системной модели вычислительное устройство представляется в виде программ, имитирующих его будущую работу. При этом важно сохранить баланс между упрощением такой модели и возможности получить на их основе схему. Например, если модель представляет собой блок, обозначенный как «искусственный интеллект, принимающий оптимальное решение», она бесполезна на практике. Впоследствии другому специалисту придется полностью разработать такой блок, о детализированной работе которого ему не представили никаких сведений. Вместо этого модели должны имитировать работу конкретных компонентов, поведение которых уже известно и может быть описано с достаточным уровнем детализации. Например, модель сумматора в виде строки a = b + c является достаточной, поскольку общие принципы реализации таких компонентов известны.

Вместе с этим системная модель должна отслеживать потенциально проблемные ситуации при последующей реализации схемы. Например, для электронных компонентов не допускается одновременная подача сигналов на вход из двух и более источников. Если представить системную модель в виде выражений a = 0; a = 1, это не является проблемой с точки зрения программиста, поскольку в этом случае процессор выполнит операции последовательно. Если же передать такую системную модель для аппаратной реализации, придется определить, что именно следует подать на вход элемента, который будет хранить переменную a. Вместо этого в системной модели необходимо описать процесс так, чтобы было понятно, что присваивания выполняются в разные моменты времени.

Для описания такого стиля моделирования часто используется понятие транзакции. Оно означает такую операцию, которая может быть выполнена только полностью. Например, процесс записи значения в ячейку памяти означает подачу на модуль памяти всех сигналов – адреса ячейки, данных для этой ячейки и сигнала, указывающего на необходимость выполнения операции записи. Все эти действия должны быть смоделированы совместно. При этом для программиста операторы addr = 123; data = 45; write = 1 выглядят независимыми действиями, однако он должен описывать модель так, чтобы все сигналы были согласованными и полностью описывали поведение моделируемого блока в определенный момент времени.

Назначением системного моделирования является раннее выяснение будущих свойств проектируемой системы. Например, операции с памятью, описанные в виде разработанных моделей read\_memory(addr, data) и write\_memory(addr, data), позволяют быстро определить, что будет храниться в памяти после выполнения определенной программы. Альтернативой была бы разработка схемы такого модуля памяти, что займет существенно больше времени.

После разработки системной модели становится понятно, какие функции должен выполнять процессор, какой объем памяти ему требуется, какие устройства должны быть к нему подключены и т.д. На основе системных моделей можно приступать к разработке электрической схемы, которая представляется на *уровне регистровых передач* (Register Transfer Level). На этом уровне схема описывается в виде фрагментов, передающих данные от одного регистра к другому с их возможным преобразованием. Например, если существуют регистры, хранящие сигналы a, b, c, то на уровне RTL-представления возможна запись a = b + c, которую можно преобразовать в конкретные цифровые компоненты.

При работе на уровне RTL-описания схемы обычно используются языки описания аппаратуры (Hardware Description Language), наиболее распространенными из которых являются VHDL (VHSIC HDL – Very High Speed Integrated Circuit HDL) и Verilog HDL (часто сокращается до Verilog в неформальном обсуждении). Тексты на этих языках преобразуются с помощью программ-синтезаторов в представление, которое называется *списком связей* (netlist).

*Рисунок 1.6. Взаимодействие основных этапов проектирования*

Можно перечислить следующие уровни проектирования:

**1. Поведенческий (behavioral level).**

На этом уровне устройства и их элементы представляются в виде абстрактных моделей. Например, сумматор может быть представлен в виде математического выражения a = b + c, не вдаваясь в технические детали того, как это выражение реализуется с помощью цифровых компонентов. На поведенческом уровне могут также использоваться выражения, моделирующие задержки распространения сигналов, однако такие задержки не рассчитываются, а принудительно задаются разработчиками, чтобы выяснить, как устройство будет работать, если его компоненты будут срабатывать не мгновенно. Например, сумматор может быть описан следующим выражением на VHDL:

a <= b + c after 2 ns;

В этом случае программы моделирования покажут изменение сигнала на выходе сумматора через 2 нс после подачи входных значений, но это будет сделано принудительно, а не по результатам моделирования какого-либо конкретного сумматора.

**2. Уровень регистровых передач (RTL, register transfer level)**

На этом уровне описание схоже с поведенческим, однако схема ориентирована на описание схем, соединяющих регистры друг с другом. На уровне регистровых передач игнорируются задержки, задаваемые разработчиком.

3. Вентильный уровень (gate level)

На уровне отдельных логических элементов (вентилей, *gate*) большая схема, реализующая некоторую логическую функцию, разбивается на отдельные вентили. Сложное выражение может не иметь готового решения в виде отдельного вентиля, поэтому программное обеспечение строит конечную схемотехническую реализацию из тех вентилей, которые есть в наличии. Например, функцию 4И можно составить из нескольких вентилей 2И.

4. Топологический уровень (cells level)

Уровень отдельных ячеек (cell), представляющих собой логические вентили, триггеры, мультиплексоры и прочие базовые компоненты цифровых схем.

5. Физический уровень (switch/mask level)

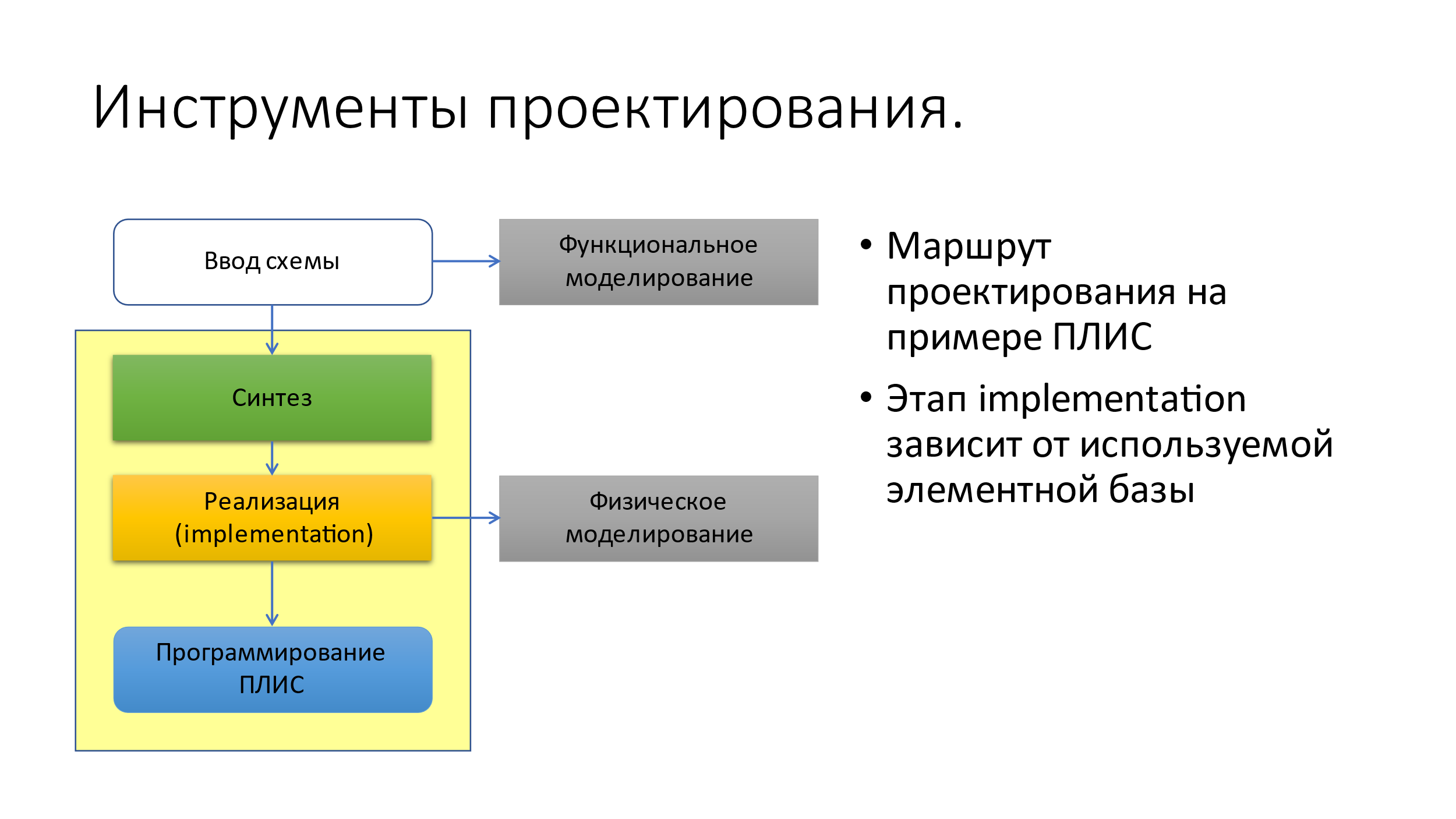
Проектирование на этом уровне предусматривает размещение отдельных транзисторов и их элементов. Обычно компоненты физического уровня поставляются в виде готовых библиотек.

1.6 Инструменты проектирования

Моделирование будущей микросхемы является важным, но не единственным шагом перед ее производством. Некорректно составленная модель может иметь скрытые логические ошибки – разработчик, неправильно понимающий моделируемые процессы, может не задать в модели требуемые сценарии проверки. Поэтому кроме компьютерного моделирования необходимо создать макет будущей микросхемы. Для этого используются программируемые логические интегральные схемы (ПЛИС), с помощью которых можно собрать цифровую схему из конфигурируемых ячеек, изменяя ее внутренние соединения по мере необходимости. Этот процесс чем-то напоминает сборку будущей пластмассовой модели из кубиков Лего. Получаемая конструкция явно не соответствует штампованной фигурке, не может реализовать какие-то сложные формы, однако дает общее представление о том, что получится при изготовлении пресс-формы для штамповки пластмассы. Эта аналогия хороша тем, что как сборку фигурки из Лего, так и программирование ПЛИС можно выполнять множество раз без потери материальных ресурсов, а изготовление пресс-формы для пластмассы требует денег и времени. Еще больше стоит подготовка производства для изготовления полупроводникового кристалла.

Кроме того, что макет на базе ПЛИС предоставляет наглядную демонстрацию работы будущего устройства, он также выявляет так называемые «эффекты реального мира». Например, частота тактовых сигналов на практике подвержена небольшим изменениям, которые трудно адекватно смоделировать в силу случайного характера процессов. Изменения сигналов на входах микросхемы, заданные моделью, также вряд ли будут происходить в те моменты, которые задаются моделью. Окажет ли это существенное влияние на работоспособность микросхемы? Умозрительно ответить на этот вопрос достаточно сложно, тем более что риск получить некорректно работающую микросхему высок, а проверить гипотезу путем создания соответствующей схемы в ПЛИС относительно недорого. Поэтому предварительное создание макета на базе ПЛИС является практически обязательным шагом при разработке новых микросхем.

Маршрут проектирования цифрового устройства на базе ПЛИС показан на рис. 1.7.



*Рисунок 1.7 Маршрут проектирования цифрового устройства на базе ПЛИС*

На рисунке видны основные этапы, показанные укрупненно. Эти этапы имеют явное соответствие в САПР. Ввод схемы происходит в основном на поведенческом или RTL уровнях с применением языков описания аппаратуры. Можно провести предварительное функциональное моделирование, анализируя только конструкции языка (без попытки привязать их к элементам ПЛИС). Этот этап можно проводить и без ПЛИС, с помощью в том числе и свободно распространяемых программ моделирования.

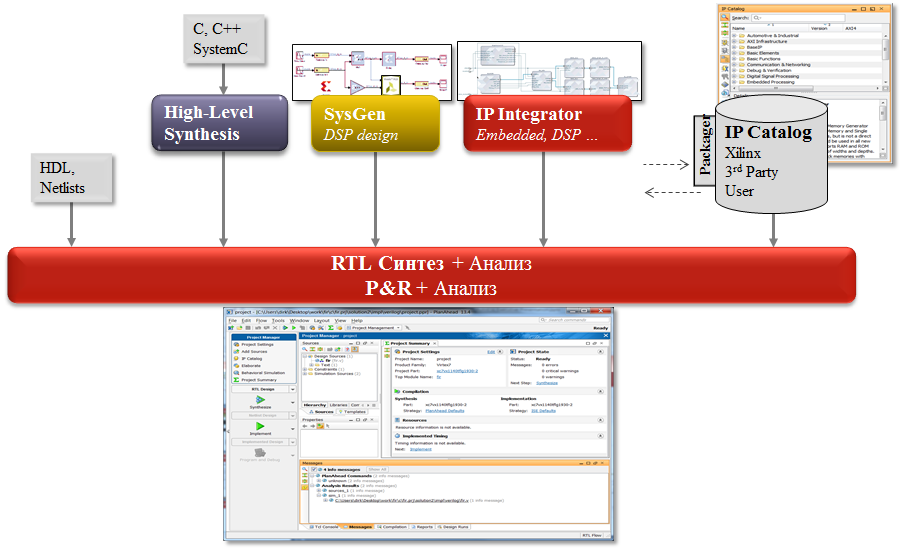
На рис. 1.8 показаны примеры отладочных плат на базе ПЛИС. Такие платы содержат дополнительные компоненты (подсистему питания, память, порты USB и Ethernet, разъемы и т.д.) и готовы к применению, обычно с подключением к компьютеру с помощью USB.



*Рисунок 1.8 Примеры отладочных плат на базе ПЛИС*

Платы, показанные на рис. 1.8, представляют собой достаточно широкий диапазон ресурсов. В простейшем варианте (верхний левый угол) показана плата, предназначенная для обучения и освоения. В нижнем ряду показан модуль компании Synopsys, предназначенный для моделирования будущих СБИС большого объема.

Поскольку ПЛИС могут использоваться и в составе самостоятельных изделий (а не только как макеты будущих микросхем), для них применяются и другие средства описания схем, кроме языков описания аппаратуры. Соотношение инструментов проектирования для ПЛИС показано на рис. 1.9. В основном дополнительные инструменты являются специализированными, и работают по принципу автоматической генерации текстов на языках описания аппаратуры.



*Рисунок 1.9 Основные инструменты проектирования на базе ПЛИС*

Среди современных инструментов проектирования можно упомянуть:

- языки высокого уровня (HLL, High Level Languages), основанные на Си (C++, System C);

- генератор систем цифровой обработки сигналов (цифровых фильтров, модулей спектрального анализа и т.п.) System Generator for DSP;

- генератор IP-ядер (готовых компонентов различного назначения).

Важным понятием является IP-ядро (IP – Intellectual Property, «интеллектуальная собственность»). Это готовый блок, который может быть использован в составе проекта, выполняя определенную функцию. У таких блоков нет четкого определения, ни по формату предоставления (схема или фрагмент кристалла), ни по назначению (блоком может быть простой сумматор или сложный контроллер). Роль IP-ядер в проектировании – предоставление готовых фрагментов для сложного проекта. Принципиальным вопросом здесь является возможность быстрой интеграции такого ядра в проект. Можно еще раз отметить, что не существует какого-либо специального формата или языка описания для IP-ядер. Иногда отмечается, что отличием IP-ядра является возможность получения технической поддержки от производителя.

Дополнительные инструменты проектирования не всегда могут быть использованы для разработки СБИС, поскольку в ряде случаев они привязаны к особенностям ПЛИС и используют специализированные компоненты. Эти компоненты не будут автоматически перенесены в СБИС на основании только предоставленного описания, поскольку в таком высокоуровневом описании будет применяться ссылка на «черный ящик», который имеется только у компании-производителя ПЛИС. При проектировании собственной микросхемы такие компоненты, как контроллер PCI Express, USB, Ethernet, блоки цифровой обработки сигналов (и ряд других) необходимо разработать самостоятельно (или приобрести лицензию на готовый компонент).

1.7 Выводы по разделу

Проектирование вычислительной техники включает в себя деятельность в нескольких смежных направлениях – разработка аппаратной составляющей, разработка программного обеспечения, исследование предметной области и др. Для упрощения взаимодействия между специалистами процесс проектирования разбивается на этапы по времени разработки, а также имеет различные уровни проектирования, от верхних уровней математических моделей до низкоуровневых деталей технологической реализации.

Важными предварительными этапами являются моделирование с помощью специального программного обеспечения (simulator) и макетирование путем загрузки разработанной схемы в ПЛИС. Этап макетирования важен, поскольку он позволяет наглядно продемонстрировать работу будущей микросхемы, к тому же выявляет возможные несоответствия модели реальному миру.

Процесс разработки на базе ПЛИС в целом соответствует процессу разработки обычной микросхемы, однако ПЛИС содержат и дополнительные компоненты, которые не будут автоматически перенесены в будущую микросхему. Поэтому макетирование необходимо проводить с учетом возможностей будущего производства и имеющихся в наличии компонентов.

Контрольные вопросы:

1. Что входит в состав компьютера в модели фон Неймана?

2. На какие компоненты можно разбить вычислительную систему?

3. Чем отличаются системная модель, уровень регистровых передач и топологическое представление?

4. Почему для разработки макета используется ПЛИС? Можно ли обойтись компьютерным моделированием схемы?