Проектирование СБИС.
Стили и этапы проекта

Удивительно, как при начале пира пьют из малых чаш,
а с полными желудками — из больших.
Анахарсис

В прошлом году завершилась публикация цикла «Школа схемотехнического проектирования устройств обработки сигналов». Я взял небольшой тайм-аут (чем слегка расстроил любимую редакцию), дабы задуматься — а что, собственно говоря, нужно современному разработчику? Результат этих размышлений уважаемый читатель может видеть, начиная с этого номера.

Владимир Стешенко

steshenk@sm.bmstu.ru

Мной продуман и начат цикл статей (в расширенном виде он, вероятно, преобразуется в книгу), посвященный вопросам проектирования устройств как на ПЛИС, так и на СБИС (сверхбольших интегральных схемах). Мне не очень нравится сей термин, но, по-моему, он определенным образом отражает понятие ASIC. В отличие от моих предыдущих работ, упор делается на алгоритмическую сторону дела. Я планирую рассмотреть общую методологию проектирования СБИС, ее основные этапы применительно к современным средствам САПР, рассказать об архитектурных особенностях арифметических и логических узлов, привести несколько конкретных примеров реализации. Я прошу всех заинтересовавшихся присылать отклики, особенно критические — критику мы любим, полемику обожаем. Итак, начнем…

Еще несколько лет назад в головах «обычных» разработчиков применяемая ими элементная база представлялась данной «от Бога». Мысли о том, что в какой-то момент придется собственноручно проектировать интегральную схему, да еще достаточно сложную и большую, могли присниться разве что в страшном сне. Считалось, что это прерогатива разработчиков интегральных схем, у них свои методы и приемы, мало интересные традиционному схемотехнику. В настоящее время ситуация несколько изменилась. Во-первых, появление такой элементной базы как ПЛИС и идеологии систем на кристалле (system-on-a-chip — SOC) наводят на мысль, что при выходе на большие серии (всегдашняя мечта!) необходим переход на более дешевую и удобную технологию. Еще один момент связан с разработкой аппаратуры, эксплуатируемой в достаточно жестких условиях — номенклатура ПЛИС, способных выдержать режимы, возникающие на борту космических летательных аппаратов, довольно узка, да и стоимость и возможность приобретения весьма и весьма затрудняют их применение. Основным фактором, сдерживающим разработку специализированных БИС малыми силами, был и пока остается фактор стоимости специализированного ПО и средств проектирования. Хотя, уже сейчас она медленно, но верно падает. В то же время перед разработчиком открываются возможности проводить значительный объем работ по проектированию СБИС, используя средства проектирования, широко применяемые при работе с ПЛИС (Leonardo Spectrum, Modelsim), естественно, при наличии соответствующих библиотек. В этом случае возможно выполнить функциональное и временное моделирование и осуществить логический синтез — дальше остается работа тополога. Темой сегодняшнего занятия и будет круг вопросов, возникающих при переходе к разработке СБИС, — конечно, материал занятия отнюдь не исчерпывающий, но самое общее представление он, наверное, дает.

Стили проектирования СБИС

Как и проектирование любого мало-мальски серьезного объекта, проектирование специализированной СБИС (Application-Specific Integrated Circuits — ASIC) начинается с определения базовых функций ее составных частей. Эта стадия важна для выбора соответствующего стиля реализации проекта (design style), который, по идее, должен быть наиболее пригодным к реализации проекта и его последующей верификации. Стиль проекта определяет соответствующие шаги в маршруте проектирования, а также используемые библиотеки. Очевидно, что грамотный выбор стиля реализации проекта позволяет определить оптимальное соотношение между характеристиками системы, ее стоимостью и сроками и объемом выпуска.

По стилю проектирования и исполнения СБИС делятся на заказные (custom) и полузаказные (semi-custom) проекты (рис. 1). Полностью заказные СБИС,

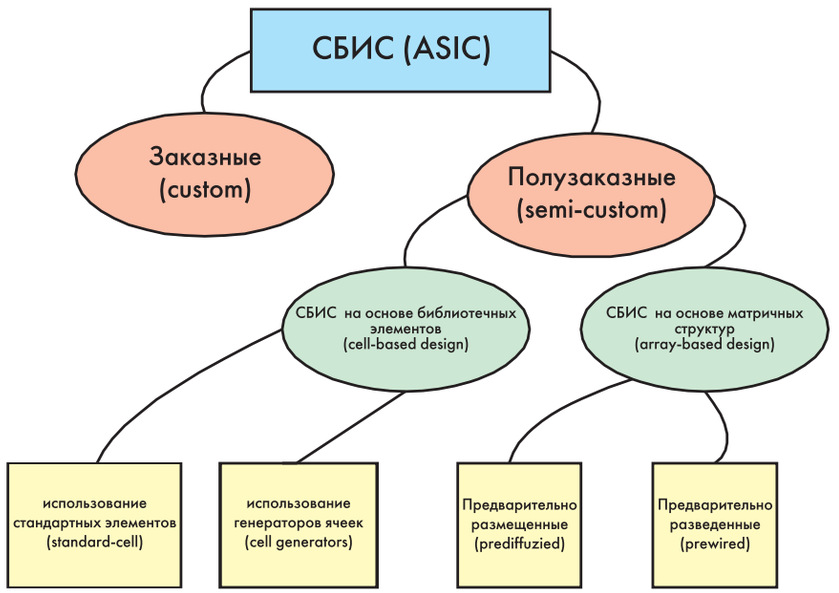

Рис. 1. Классификация стилей проектирования БИС

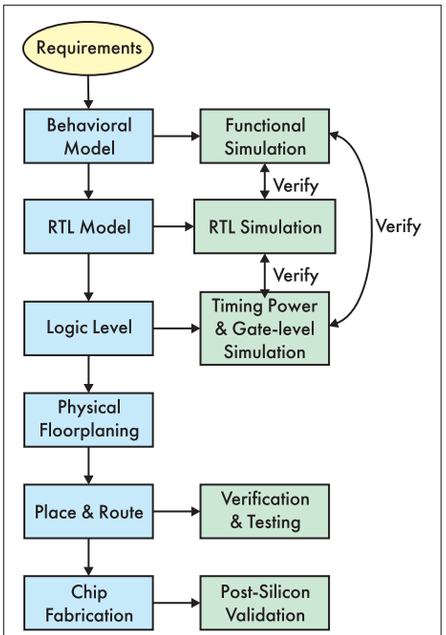

Рис. 2. Типичный маршрут проектирования

как следует из названия, представляют полностью выполненный законченный проект, обеспечивающий максимальную производительность и низкую цену, но только при крупносерийном производстве. Стоимость же разработки и отладки очень высока. Особенно это касается моментов верификации (как на этапе программной, так и аппаратной). Кроме того, полностью заказные СБИС имеют самый большой срок разработки.

Полузаказные СБИС имеют различного рода ограничения на используемые библиотеки, в них широко используются так называемые ядра интеллектуальной собственности (IP-cores), в то же время разработчик лишен возможности «долизывать» (fine-tuned) компоненты таких СБИС для достижения экстремальных характеристик, впрочем, на практике это и не требуется. Как правило, предварительно разработанные и разведенные блоки имеют хорошие характеристики в отношении производительности и занимаемой площади на кристалле, но самое важное — они позволяют поднять уровень абстракции при описании проекта. Очевидно, что использование таких блоков позволяет резко ускорить появление новых образцов БИС на рынке. Как известно, блоки интеллектуальной собственности активно поддерживаются соответствующими средствами САПР.

Полузаказные СБИС можно классифицировать на СБИС на основе библиотечных элементов (cell-based design) и БИС на основе матричных структур (array-based design).

При проектировании СБИС на основе библиотечных элементов используют соответствующие библиотеки предварительно разведенных библиотечных компонентов (cells) или специализированные генераторы таких элементов (cell generators), например модулей памяти, которые формируют трассировку элемента по его функциональному описанию.

Разработка на базе библиотечных элементов в свою очередь подразумевает либо использование стандартных элементов (standard-cell) , либо использование генераторов ячеек (cell generators) для реализации примитивов. Традиционно генераторы используются для синтеза ячеек памяти, массивов программируемой логики (programmable logic arrays, PLA), сложных устройств распределения и обработки потоков данных (datapath components), таких, как перемножители, мультиплексоры и т. п. Генераторы ячеек полностью параметризованы и могут обеспечить, например, различную разрядность устройств, типы памяти и т. д.

В отличие от БИС на основе библиотечных элементов, полузаказные БИС на основе матричных структур представляют предварительно размещенные, но не соединенные базовые логические элементы, расположенные в виде матрицы. К таким БИС относятся, соответственно, базовые матричные кристаллы, масочные и лазерно-программируемые ПЛИС (MPGA, LPGA), а также перепрограммируемые структуры ПЛИС (FPGA на основе технологий SRAM и antifuse), достаточно подробно рассмотренные в литературе.

Разработка прототипов и верификация схем на базе технологий FPGA последние годы стала очень популярной из-за невысокой цены при малом количестве производимых изделий.

Рассмотрим типичный маршрут проектирования СБИС (рис. 2).

Маршрут проектирования (design flow) определяет этапы проектных процедур, используемых на всех стадиях разработки, — от выработки и формализации идеи до тестирования готовых образцов.

Традиционно при проектировании специализированных БИС используется нисходящая модель маршрута проектирования (waterfall model). При такой организации маршрута проектирования проект проходит различные фазы, постоянно увеличивая детализацию представления. Нисходящее проектирование подразумевает минимальное взаимодействие между командами разработчиков на различных фазах проекта. Процесс проектирования начинается с разработки технических требований (specification), их последующего анализа, проведения предварительного моделирования с помощью специализированных пакетов или на языке высокого уровня (например, C).

Здесь хотелось бы отметить, что, несмотря на широкий набор инструментов моделирования, при проектировании СБИС для обработки сигналов задача моделирования усложняется необходимостью разработки не только модели системы, но и модели тестовых воздействий с учетом шумов, эффектов квантования и особенностей тракта. На выходе первого этапа должна быть осуществлена полная функциональная проверка технических требований.

На следующем этапе осуществляется описание проекта с помощью одного из языков описания аппаратуры, как правило VHDL или Verilog, на уровне регистровых передач (register transfer level, RTL).

Функциональные возможности описания на уровне регистровых передач моделируются и верифицируются относительно исходных технических требований (например, модель на C или в MatLAB), которая используется как эталонная модель (golden model) для верификации проекта на каждом уровне абстракции. Данный этап и называется функциональной верификацией модели.

По описанию на уровне RTL с помощью программы логического синтеза формируется список цепей (gate level netlist), учитывающий задержки на библиотечных элементах (но, как правило, не учитывающий временные задержки на межсоединениях), который используется для временной верификации проекта (timing verification). Цель временного моделирования — проверить, удовлетворяет ли разрабатываемая БИС заданным временным ограничениям (timing constraints).

На основании данных синтеза топологи (physical design team) разрабатывают и оптими-

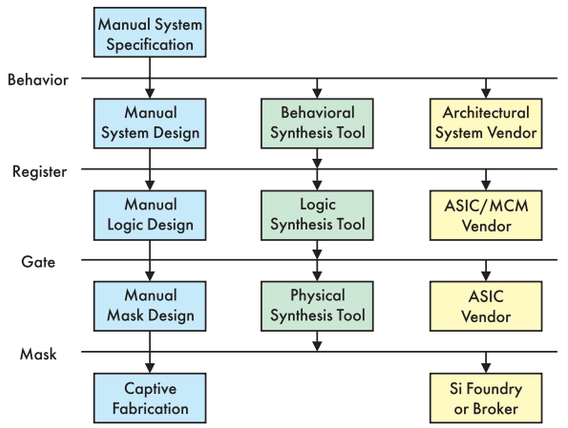

Рис. 3. Сравнение методологий проектирования

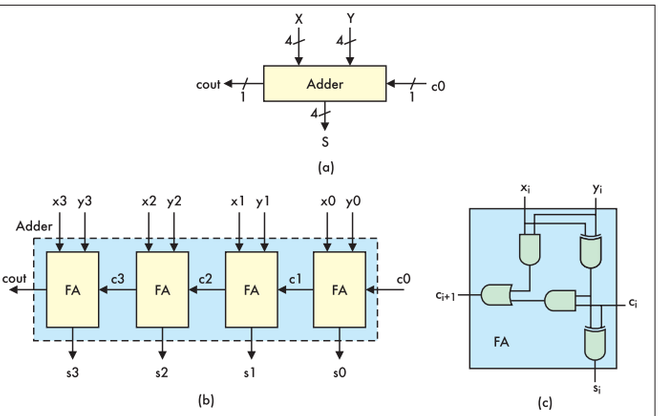

Рис. 4. Иерархическое деление проекта возможно в двух направлениях — снизу-вверх и сверху-вниз

зируют разводку кристалла (floorplan), размещая библиотечные элементы и межсоединения неким оптимальным образом. После разработки топологии можно повторно выполнить формирование файла задержек и последующее временное моделирование, учитывающее влияние межсоединений. Затем кристалл можно передавать в производство и осуществлять последующее тестирование образцов.

Недостаток этой методологии проектирования — с увеличением сложности проекта увеличивается опасность появления ошибок, и затрудняется процесс их поиска. Более того, насколько удовлетворяет разрабатываемая БИС предъявляемым к ней требованиям, становится ясно только в самом конце процесса проектирования. Ошибки, обнаруженные в конце той или иной стадии проектирования ведут к повторному ее выполнению, что в ряде случаев влечет неоднократный выпуск прототипов (shuttles), значительно замедляя сроки выполнения проекта и резко повышая его стоимость.

Технические требования к проекту представляются его поведенческой моделью (behavioral model), которая определяет временные ограничения, ограничения по площади кристалла и потребляемой мощности, тестопригодность и т. п. Такая поведенческая модель обычно задается в форме выполнимых функциональных описаний на языке типа C (или C++). По этим функциональным описаниям затем выполняется моделирование для широкого набора входных воздействий.

Например, при разработке нового микропроцессора после выбора общей архитектуры производится разработка структуры системы команд. Определяется уровень конвейеризации, разрядность адреса и данных, параметры и типы используемых регистров и т. п. Параллельно разрабатывается симулятор системы команд, чтобы можно было осуществить проверку ее эффективности и начать разработку программного обеспечения параллельно с разработкой самого процессора. В этом случае без программного имитатора процессора не обойтись, он позволит отловить ошибки в архитектуре процессора, в частности ошибки в организации конвейера. Кроме того, использование программного имитатора позволит отладить систему команд и внести в нее необходимые изменения.

Переход от модели на функциональном или поведенческом уровне к описанию на уровне регистровых передач осуществляется либо вручную, написанием соответствующего кода на языке описания аппаратуры, либо с использованием специализированных средств синтеза высокого уровня (high-level synthesis tool). В частности, такой продукт, как Systemview фирмы Elanix, позволяет получить описание на VHDL из функциональной модели.

Описание модели на уровне регистровых передач использует компоненты типа сумматоров, перемножителей, регистров, мультиплексоров и т. п., чтобы представить структуру проекта и его межсоединения. Описание на уровне RTL моделируется, как правило, выполняется событийное моделирование (event-driven simulation) с целью верификации функциональности и основных временных характеристик. Верифицированная функциональная модель служит основой для синтеза на уровне логических вентилей (библиотечных компонентов).

Логический синтез представляет собой методологию проектирования для оптимизации на уровне логических элементов (gate-level).

До появления методологии логического синтеза разработчики СБИС использовали методологию схемного описания и последующего моделирования. При использовании этой методологии разработка начинается с создания структурной схемы кристалла. Затем на основании структурной схемы создавалась принципиальная схема устройства с использованием соответствующих средств САПР. После этого выполнялась трассировка кристалла и его производство.

При методологии логического синтеза описание создается на одном из языков описания аппаратуры, как правило, на VHDL или Verilog. При описании проекта используются Булевы уравнения, модели на уровне конечных автоматов.

На рис. 3 показано сравнение этих методологий проектирования.

В настоящее время большинство фирм-разработчиков БИС являются так называемыми фаблесс-компаниями (то есть не имеют собственной производственной базы). В этом случае изготовление кристаллов осуществляется на мощностях специализированных кремниевых фабрик, которые предоставляют разработчикам библиотеки для логического синтеза. Собственно специалисты фабрик осуществляют окончательную доработку фотошаблонов и изготовление кристалла. В этом случае актуальным становится использование блоков интеллектуальной собственности (intellectual property — IP), которые представляют собой полностью отработанные и разведенные элементы, как правило, используемые в системах на кристалле.

Для успешного выполнения любого сложного проекта необходимо организовать его иерархическую декомпозицию — выделить простые составные части. На рис. 4 приведен пример декомпозиции 4-битового сумматора, который состоит из четырех однобитных сумматоров, в свою очередь каждый однобитный сумматор состоит из набора вентилей.

Другой иерархический подход базируется на концепции абстракции проекта. В процессе проектирования выделяются различные уровни абстракции в зависимости от стадии проектирования — от идеи до производства. Так, на рис. 5 показано, что в зависимости от уровня представления объектом абстракции является система, регистр,

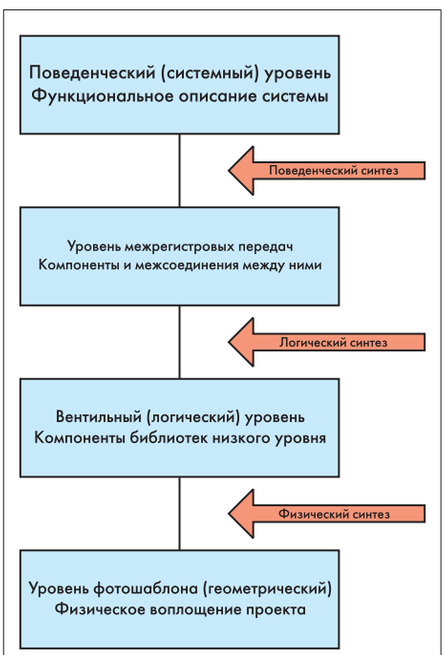

Рис. 5. Поведенческое описание

вентиль, геометрия библиотечного элемента на кристалле.

Системный уровень описания (system-level description) проекта состоит из поведенческого описания в терминах функций, выражений, алгоритмов. На уровне регистровых передач (register transfer level) проект представляется совокупностью арифметических и логических узлов, элементов памяти и т. п. Вентильный или логический уровень (logic level) описывает проект на уровне логических вентилей (logic gates) и триггеров (flip-flops). В этом случае поведение схемы может быть описано системой логических уравнений. Эти логические элементы представляются на кремниевом (топологическом) уровне (geometric level) в виде топологических элементов и межсоединений.

На рис. 5 представлено поведенческое описание как начальный уровень абстракции, который представляет функциональные возможности проекта на системном уровне. Уровень регистровых передач включает компоненты и межсоединения между ними, для большего количества сложных систем может также включать типовые элементы типа ПЗУ, СБИС. Вентильный (логический) уровень соответствует представлению уровня логического элемента, и набор шаблонов топологических элементов кристалла соответствует геометрическому уровню. Следует обратить внимание на следующие моменты, показанные на рис. 5. Во-первых, на рисунке показаны основные проектные процедуры и используемые средства САПР, в зависимости от уровня представления проекта и, соответственно, уровня детализации. Во-вторых, представленный процесс синтеза состоит из процессов поведенческого синтеза (behavioral synthesis), логического синтеза (logic synthesis) и физического синтеза топологии (physical synthesis). В дальнейшем изложении мы рассмотрим эти этапы подробнее.

Различные уровни представления проекта различаются типом информации, которую они отображают. Поэтому уровни представления могут быть классифицированы как поведенческий, структурный и физический.

В поведенческом представлении описано только функциональное поведение системы, и проект представляется как «черный ящик», имеющий зависимость выходного сигнала от входного. Структурное представление детализирует проект, вводя информацию относительно компонентов в системе и их взаимодействия. Детальные физические характеристики компонентов определены в физическом представлении, включая информацию о размещении и трассировке.

Зависимости между различными уровнями абстракций и представлениями проекта отображены на Y-диаграмме (рис. 6). Эта диаграмма показывает, как тот же самый проект на системном уровне может иметь поведенческое представление и структурное представление. Принимая во внимание, что поведенческое представление описывает проект в терминах блок-схем и алгоритмов, структурное представление представляет проект в терминах процессоров, блоков памяти и других логических блоков. Точно так же поведенческое представление на уровне межрегистровых пересылок представило бы поток межрегистровых пересылок набором поведенческих инструкций, а структурное представление представляет тот же самый поток набором компонентов и связей между ними. На логическом уровне схема может быть представлена булевыми уравнениями или конечными автоматами в поведенческом представлении, либо как цепь связанных вентилей и триггеров в структурном представлении. Геометрический уровень представлен как транзисторные функции в поведенческом уровне, как микротранзисторы в структурном представлении, а также как топология, ячейки, кристаллы в физическом представлении. Таким образом, модель Y-диаграммы помогает понять различные стадии и степень подробности для представлений проекта. Можно расширить эту модель, добавив аспекты верификации модели и испытания готового кристалла.

Как известно, любой проект начинается с определения системных требований. Обычно это осуществляется в форме документа технических требований. Эти технические требования описывают требования к конечному изделию, функциональные возможности и другие требования типа температурного диапазона, потребляемой мощности, требований приемки пользователя и системного испытания. Это ведет к более определенным требованиям на устройство непосредственно, в терминах функциональных возможностей, интерфейсов, рабочих режимов, условий эксплуатации, эффективности, отражаемых в техническом задании.

На этой стадии начальный анализ выполняется на основе системных требований, чтобы определить выполнимость технических требований, какой стиль проектирования будет использоваться, фабрику (foundry), на которой планируется выпуск, технологический процесс, библиотеки и т. п. Некоторые другие параметры типа вида корпуса, рабочей частоты, числа контактных площадок на кристалле, площади, размера и вида используемой памяти также оцениваются на этом этапе. Традиционно, для простых проектов ввод проекта выполняется после того, как проект архитектуры более высокого уровня будет закончен. Ввод проекта может быть в форме схемных решений блоков, которые реализуют выбранную архитектуру. Однако с увеличивающейся сложностью проектов соображения относительно системного моделирования и инструментальных средств проверки становятся преобладающими. Системные проектировщики хотят гарантировать, что аппаратные средства ЭВМ проектируют качественно и быстро создают рабочую аппаратную модель, моделируют ее взаимодействие с остальной частью системы, осуществляют синтез и формальную верификацию. Следовательно, в качестве средства ввода проекта используют языки описания аппаратуры высокого уровня (hardware description languages — HDL) для задания начальных технических требований системы. В существующих методологиях проектирования специализированных интегральных схем, используемых в промышленности, языки описания аппаратуры обычно используются, чтобы описывать проекты на уровне межрегистровых пересылок. Однако в последнее время стала пользоваться популярностью методология «Описал — выполнил — долизал» (Specify-Explore-Refine — SER). После этапа постановки задачи (спецификации исходных требований), на стадии реализации происходит оценка различных элементов системы, чтобы осуществить системные функциональные возможности в пределах указанных конструктивных ограничений. Технические требования модифицируются на стадии доводки проекта в соответствии с проектными решениями, осуществленными на стадии реализации.

Эта методология ведет к лучшему пониманию функциональных системных возможностей на очень ранней стадии в процессе проектирования. Выполнимость технических требований особенно полезно проверить на правильность функциональных возможностей изделия и пригодность для автоматической проверки. Рабочие технические требования могут легко моделироваться, и та же самая модель может использоваться для синтеза. Обычно производят проверку функциональных моделей на языках C или C++ после завершения моделирования, проект вручную снова вводится в инструментальные средства САПР с использованием языков описания аппаратуры.

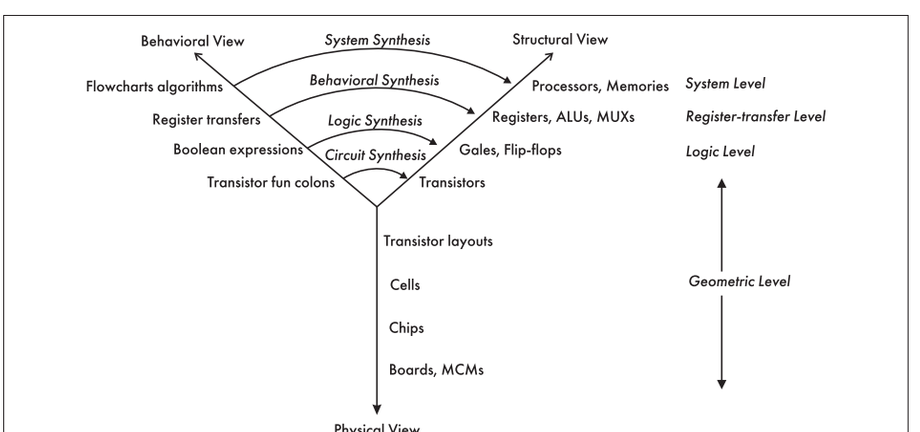

Рис. 6. Зависимости между различными уровнями абстракций и представлениями проекта

Таким образом, выбор языка для ввода описания системы является областью самых активных дискуссий и исследований. Недаром вопросы, какой из языков описания лучше, не сходят с конференций в Интернете как у нас в стране, так и за рубежом. Язык должен быть прост для понимания и программирования и должен быть способен отображать характеристики всей системы, а также поддерживаться инструментальными средствами автоматизированного проектирования. Известны такие языки описания аппаратуры, как VHDL, Verilog, HardwareC, Statecharts, Silage, Esterel и Specsyn.

Имеется тенденция к использованию в качестве языков описания аппаратуры языков программирования из-за возможности легко описать поведение и выполнить моделирование, а также из-за знакомства проектировщиков с языками программирования высокого уровня общего применения, типа C и C++. Эти языки подняли уровень абстракции, при которой проектировщик определяет проект ближе к концептуальной модели. Тогда концептуальный поведенческий проект может быть разбит на разделы и структурирован, входящие в него компоненты могут быть распределены. При таком подходе проект прогрессирует от вполне функциональных технических требований до структурной реализации за несколько последовательных шагов. Эта методология приводит к снижению времени разработки, более эффективному использованию большего пространства проекта и позволяет снизить время перепроектирования (re-design time).

Мы рассмотрели основные вопросы определения стилей проектирования и его этапов. В следующей статье цикла будут рассмотрены вопросы поведенческого синтеза, тестопригодности проекта, логического синтеза и физического воплощения.