Проектирование в условиях временных ограничений: компиляция проектов

Статья продолжает рассмотрение вопросов создания проектов на основе ПЛИС с условием максимального ускорения процедуры проектирования. Средства обнаружения дефектов, рассмотренные в предыдущих выпусках серии публикаций, оказываются малоэффективными, если время устранения дефекта будет значительным. Довольно большая составляющая этого времени связана с повторной компиляцией проекта. Этот материал содержит как общие соображения по сокращению временных затрат, так и примеры практической работы с компилирующими средствами фирмы Altera.

Ростислав ГРУШВИЦКИЙ
RIGrushvitsky@mail.eltech.ru
Максим МИХАЙЛОВ
yamaksya@yandex.ru

Проблема ускорения процессов компиляции

«Лучше день потерять,
а потом за пять минут долететь»
м/ф «Крылья, ноги и хвосты»

Как уже говорилось в предыдущих выпусках, в существующих условиях время выхода на рынок с готовой продукцией (не только в форме FPGA, но и в виде ASIC) не в последнюю очередь определяется временем отладки проекта. Сокращение временных затрат на итерационную процедуру отладки проектов связано с уменьшением времени выполнения следующих этапов:

- принятие решения о схемотехнических или топологических изменениях проекта;
- внесение требуемых изменений;
- компиляция проекта.

Время, затрачиваемое на принятие решения о целесообразных изменениях проекта, зависит не столько от его сложности, сколько от профессиональной подготовленности разработчиков. Хорошие решения, как правило, требуют существенных интеллектуальных и временных затрат. Большинство проектных изменений в подобных ситуациях вносится для локализации дефекта. Интегрально время, затрачиваемое на всю процедуру отладки, определяется скорее числом итераций этой процедуры, чем непосредственно временем принятия того или иного решения. Поэтому важна эффективность выполнения каждого этапа изменения проекта. Однако поскольку изменения зачастую оказываются недетерминированными и слабо поддаются автоматизации, то трудно говорить о минимизации продолжительности данного процесса в общем случае. Этап кодирования проекта (внесение изменений) также представляется малоинтересным. В противоположность первым двум фазам особенностью заключительного этапа процедуры модификации проекта является малая зависимость от действий человека. В качестве ключевого момента здесь выступают применяемые технические методы и средства. Вместе с тем на компиляцию проектов (даже при использовании современных высокопроизводительных компьютеров) могут иногда требоваться не минуты, а часы.

Ускорение компиляции проектов на FPGA происходит по нескольким направлениям. Во-первых, можно отметить рост производительности вычислительных средств, используемых при проектировании. Это возрастание связано как с чисто количественными изменениями (повышением тактовой частоты ЦП, увеличением объемов оперативной памяти), так и с качественным (использование многопроцессорных ядер и возможность распараллеливания работы САПР). Во-вторых, улучшаются алгоритмы работы САПР. Но эти успехи нивелируются постоянным возрастанием сложности проектов.

В таких условиях проблема ускорения процесса компиляции более чем актуальна. Материал настоящей статьи посвящен рассмотрению как потенциальных вариантов ускорения компиляции проектов, так и конкретных действий разработчика для подобного ускорения.

Сплошной поток компиляции

В типовой процедуре компиляции практически все фирмы — производители САПР для ПЛИС — выделяют следующие этапы:

- анализ исходных данных проекта;
- синтез проекта (как правило, сначала на RTL-уровне, а затем и на технологическом уровне ПЛИС);
- монтирование проекта (размещение и разводка топологических элементов выбранной ПЛИС);
- сборка проекта (формирование загрузочного или программного файла);
- временной анализ проекта.

При успешной компиляции разработчик может переходить к программированию (конфигурированию) выбранной интегральной схемы.

Конечно, выделение этих этапов условно, но именно их наличие и последовательность выполнения предусмотренных в них действий одинаковы для САПР различных фирм. Подобный поток компиляции может быть назван сплошным. В этом случае приведенная процедура выполняется как при первоначальной компиляции, так и при повторных трансляциях, обусловленных исправлениями проекта при поиске и/или обнаружении дефектов проекта. Независимо от того, в какой его части произошли изменения, при сплошной компиляции происходит обработка всего исходного кода (на стадии анализа и синтеза) и переразмещение всей логики (на этапе монтирования). Даже при незначительных изменениях в одном элементе проекта процесс компиляции повторяется полностью, что влечет за собой существенные временные затраты. Однако этот недостаток сплошной компиляции в то же время оказывается ее главным достоинством. Такой подход позволяет добиться оптимального использования логических ресурсов и улучшения временных характеристик за счет глобальной оптимизации проекта.

Возможности ускорения компиляции

Естественно, что ускорение процесса компиляции (при сохранении качества) можно достичь только при учете результатов предыдущих действий. Хотя пропуск тех или иных этапов целиком возможен, но применяют его редко. Например, изменение ограничений или правил размещения и разводки может не требовать изменений списка цепей логического описания проекта. Большее распространение может получить такой вариант, как разбиение проекта на отдельные части, повторная обработка которых на определенных этапах компиляции может быть пропущена. Как следует из последовательности действий при обычной компиляции, разбиение на части можно выполнять перед различными этапами. Объединение также возможно после различного числа пройденных и пропущенных этапов.

Сама по себе идея фиксации отдельных фрагментов проекта не нова и уже более 20 лет активно используется в современных компиляторах программных продуктов. Однако в последнем случае для англоязычного термина «incremental» вместо термина «инкрементальная» обычно выбирается термин «инкрементная компиляция». Реализация этих же подходов для компиляторов аппаратных продуктов требует других решений и существенных затрат, и ее только начинают внедрять фирмы-производители ПЛИС и/или САПР для них.

Выделение фрагмента проекта в форме отдельного элемента (как правило, отдельного файла) создает возможность для его оформления в виде отдельной части (логического раздела). После выполнения компилятором анализа и синтеза в базе данных проекта образуется список внутренних и внешних цепей этого фрагмента. Этот список цепей обычно носит название послесинтезного (Post-Synthesis). Сохранение и повторное использование Post-Synthesis списка цепей — один из возможных путей ускорения процесса компиляции проекта в целом. Конечно, этот вариант возможен, если корректировка будет требовать изменений только размещения технологических элементов или разводки их межсоединений.

Выполнение следующего этапа компиляции (размещения и трассировки соединений технологических элементов ИС) над отдельным фрагментом проекта позволяет создать отдельный физический раздел проекта. Список цепей физических разделов проекта можно назвать технологическим списком цепей (Post-Fit). Сохранение и повторное использование Post-Fit списков цепей при монтировании единого проекта — другой возможный путь ускорения процесса компиляции проекта в целом. Топологическая форма физических разделов вряд ли может быть сложной (наиболее приемлемый вариант — простой прямоугольник). Естественно, что простая форма топологических разделов и блокировка их изменений приводит к отсутствию оптимальности использования логических ресурсов ИС, планируемой к монтированию проекта.

Подобное разбиение проекта на логические и физические разделы позволяет применять в современных САПР различные стратегии ускорения процесса компиляции. В общем случае безразлично, результаты какого этапа компиляции будут использоваться повторно (без их изменений). Более того, не важно, кто выполнял эту предварительную работу — проектировщик или сторонняя фирма. Объединение при разбиении на логические разделы можно осуществить как путем объединения списков цепей отдельных логических разделов (выигрыш соответствует одному этапу пропуска неизмененных разделов), так и путем объединения физических разделов (выигрыш от пропуска двух этапов обработки неизмененных разделов).

Фирмы, поставляющие САПР для компиляции проектов на ПЛИС, уделяют большое внимание вопросам ускорения процессов компиляции [1, 2]. Вместе с тем традиционно существует различие архитектур ПЛИС, выпускаемых различными фирмами, например, Xilinx и Altera, что предопределило использование ими различных стратегий ускорения процессов компиляции. Как уже отмечалось, одно из потенциальных направлений заключено в таком проектировании САПР, при котором возможно распараллеливание процесса компиляции. Мелкозернистая структура FPGA фирмы Xilinx приводила к достаточно продолжительному времени компиляции проектов. Это заставило фирму выпускать свои САПР со свойствами распараллеливания процесса компиляции между компьютерами, объединенными в локальную сеть. Xilinx предлагала запускать компиляцию сложных проектов перед выходными днями и не выключать на это время компьютеры сети. Фирма Altera стала предусматривать распараллеливание работы САПР (внутри одного ПК) между параллельно работающими процессорами лишь после повсеместного распространения многоядерных процессоров.

У большинства схем фирмы Xilinx, в отличие от ИС Altera, свойства фрагментов могут существенно изменяться от трассировки к трассировке. Чтобы избежать этого, САПР фирмы традиционно стремятся позволить проектировщику сохранять ранее полученную трассировку отдельных модулей для повторных компиляций. Проекты при реализации в ИС фирмы Altera изначально в значительно меньшей степени были подвержены влиянию изменений месторасположения отдельных элементов. Однако сближение архитектур ПЛИС двух компаний привело к тому, что Altera давно внедрила в свой САПР Quartus II средство работы с топологическими регионами под названием LogicLock, но лишь сравнительно недавно подключила к ним средства включения логических и физических разделов.

Цель авторов статьи — ознакомить читателей с основными возможностями ускорения процесса компиляции современными САПР на примере продукции конкретной фирмы. Распространенность в России разработок на ПЛИС компании Altera сделала целесообразным рассмотрение приемов ускорения процессов компиляции в САПР Quartus II. Поскольку материал базируется на возможностях, которые предоставляет последняя (на момент написания статьи) версия Quartus II ver.7.2, необходимо учитывать, что предыдущие версии САПР могут не поддерживать некоторые свойства.

Возможности ускорения компиляции в САПР компании Altera

Стандартная последовательность действий при компиляции проекта у фирмы Altera в САПР Quartus II состоит из следующих ключевых моментов:

1. Анализ и синтез (Analysis&Synthesis). На этом этапе выполняется логический синтез файлов проекта с целью минимизации логических схем и реализации проекта с учетом имеющихся ресурсов (логических элементов, трасс, блоков ввода/вывода и т. д.). Здесь же создается база данных, интегрирующая все сведения о проекте.

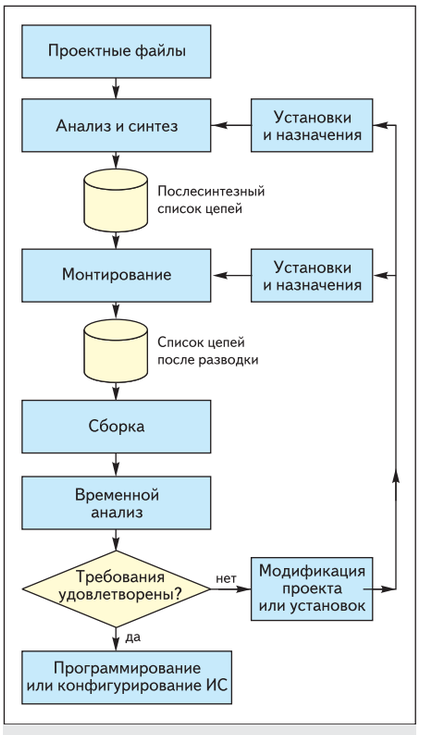

Рис. 1. Схема процесса сплошной компиляции

2. Монтирование (Fitting). Размещение и трассировка проекта в выбранном устройстве с учетом установок (settings), заданных пользователем, а также различного рода назначений (assignments).
3. Сборка (Assembling). Создание файла конфигурации устройства на основе результатов размещения и трассировки.
4. Временной анализ (Time Analysis). Анализ временных характеристик проекта.

Структурная схема сплошного потока компиляции для САПР компании Altera представлена на рис. 1.

Быстрая (smart) компиляция

Один из способов снижения временных затрат на компиляцию проекта — использование так называемой быстрой компиляции (smart compilation). Подобный перевод англоязычного термина *smart* в данном контексте, по мнению авторов, более точно отражает главную идею процесса, чем перевод, ранее введенный Д. Иоффе [3]. При включении данной опции (*Settings > Compilation Process Settings > Use Smart Compilation*) САПР определяет необходимые этапы компиляции на основе тех изменений, которые были произведены с момента последней компиляции. Этапы, не затронутые изменениями, попросту пропускаются. Например, если исходные файлы проекта остались прежними (изменения коснулись только назначений этапа монтирования проекта), то при быстрой компиляции будет пропущен этап анализа и синтеза. Однако в отличие от других методов ускорения компиляции здесь может быть пропущен только целый этап.

Инкрементальная компиляция

Более утонченные и эффективные варианты ускорения компиляции в САПР (независимо от способа ускорения) достигаются при применении методов инкрементальной компиляции.

Использование инкрементальной компиляции предполагает предварительное логическое и/или физическое разбиение проекта на несколько частей (разделов). Основное значение такого вида компиляции заключается в применении процедур трансляции только к модифицированным частям проекта, в то время как результаты компиляции неизменившихся разделов сохраняются. На заключительном этапе происходит слияние итогов предыдущих и новых компиляций.

При традиционном подходе использование единственного списка соединений (netlist) приводит к необходимости перекомпиляции всего проекта всякий раз после внесения изменений. При включенной же опции инкрементальной компиляции Quartus II может производить синтез (Synthesis) и монтирование (Fitting) каждого раздела проекта в отдельности, объединяя затем результаты в список соединений для обработки на следующих этапах процесса компиляции. Возможность многократного использования ранее полученных результатов приводит к уменьшению итерационно повторяемых этапов перекомпиляции и тем самым к сокращению общего времени отладки. Несмотря на совпадение в основе возможных вариантов инкрементальной компиляции, ее модификации оказываются востребованными при различных стратегиях (условиях) проектирования. Необходимость предварительного разбиения проекта на отдельные разделы показывает важность предварительных этапов проектирования, на которых, в частности, осуществляется выбор границ отдельных разделов, и, как правило, определяется стратегия проектирования и объединения этих разделов. Конечно, фиксация содержимого отдельных разделов может снижать возможности САПР по оптимизации проекта (недопустимость изменений реализации интерфейса между отдельными разделами), однако оптимизация самих фрагментов зачастую позволяет получить существенный выигрыш для проекта в целом.

Тем самым инкрементальная компиляция, реализованная в САПР Quartus II, позволяет:

- сохранять неизменной логическую схему при проведении изменений в другой части проекта;
- снижать время итерации приблизительно на 60%, что способствует более эффективному распределению временного ресурса;
- упрощать процедуру компиляции за счет графического интерфейса пользователя (GUI, Graphical User Interface);
- допускает управление при помощи скриптов Tcl, командной строки или сборочного файла проекта (make-файла);
- поддерживает модульную иерархию и командную разработку в соответствии с методами проектирования «сверху-вниз» и «снизу-вверх».

Инкрементальный поток компиляции с разделением проекта на составные части

В большинстве случаев инкрементальная компиляция оказывается более эффективной, нежели стандартная процедура. Основное свойство инкрементальной компиляции — это разбиение проекта на блоки: логические или физические разделы (design partitions). При успешном размещении в компоновочном плане кристалла разделов проекта удается одновременно с уменьшением временных затрат на компиляцию сохранить или даже улучшить достигнутые результаты проектирования. Допустимость разбиения единого проекта на отдельные разделы позволяет применять модификации инкрементальной компиляции при различных стратегиях проектирования. Каждый вариант стратегии оказывается более предпочтительным при определенных условиях проектирования. Рассмотрим далее модификации инкрементальной компиляции в зависимости от подобных условий.

Доработка одного или нескольких конкретных блоков

Одним из ярких примеров использования инкрементальной компиляции служит ситуация, когда большая часть проекта уже завершена и требуется доработка только одного конкретного блока. В этом случае может потребоваться сохранение характеристик немодифицируемых разделов и сокращение времени компиляции для последующих итераций. Структурная схема такого «полного» инкрементального потока компиляции приведена на рис. 2.

В таком потоке сначала проект разбивается на несколько частей. Затем на этапе анализа и синтеза выполняется логический синтез и технологическое отображение (technology mapping) каждого раздела в отдельности.

На основании назначений проекта (Design Partition Assignments) компилятор формирует один логический раздел верхнего уровня иерархии (Partition Top) и заданное число разделов нижнего уровня (Partition 1 и Partition N). Каждый из разделов отдельно обрабатывается на этапе анализа и синтеза. В том случае, если внутри раздела были произведены проектные изменения, происходит трансляция. Если же изменений не было, то список соединений для данного раздела остается прежним. В результате по завершении этапа анализа и синтеза создан индивидуальный список соединений (post-synthesis netlist) для каждого раздела.

На следующем шаге (Partition Merge) из отдельных списков соединений разделов собирается единый полный список соединений проекта. В зависимости от используемого для каждого раздела типа списка соединений в слиянии могут быть задействованы результаты синтеза (Post-Synthesis netlist), монтирования (Post-Fit netlist) или импорта низкоуровневых проектов (Imported netlist).

Далее совмещенный список соединений проходит обработку модулем монтирования (Fitter). При этом переразмещаются только те разделы, которые были изменены. В итоге формируется список всех соединений проекта для дальнейшего использования на этапах временного анализа и генерации программных файлов. Кроме того, создаются отдельные списки соединений каждого раздела для использования на этапе слияния (Partition Merge).

Возможность задавать различные (индивидуальные) опции компиляции и трассировки для тех или иных разделов и проекта в целом позволяет обрабатывать отдельные блоки с использованием опций расширенной оптимизации, в то время как остальная часть проекта остается неизменной в ходе последующих компиляций. Таким образом, до-

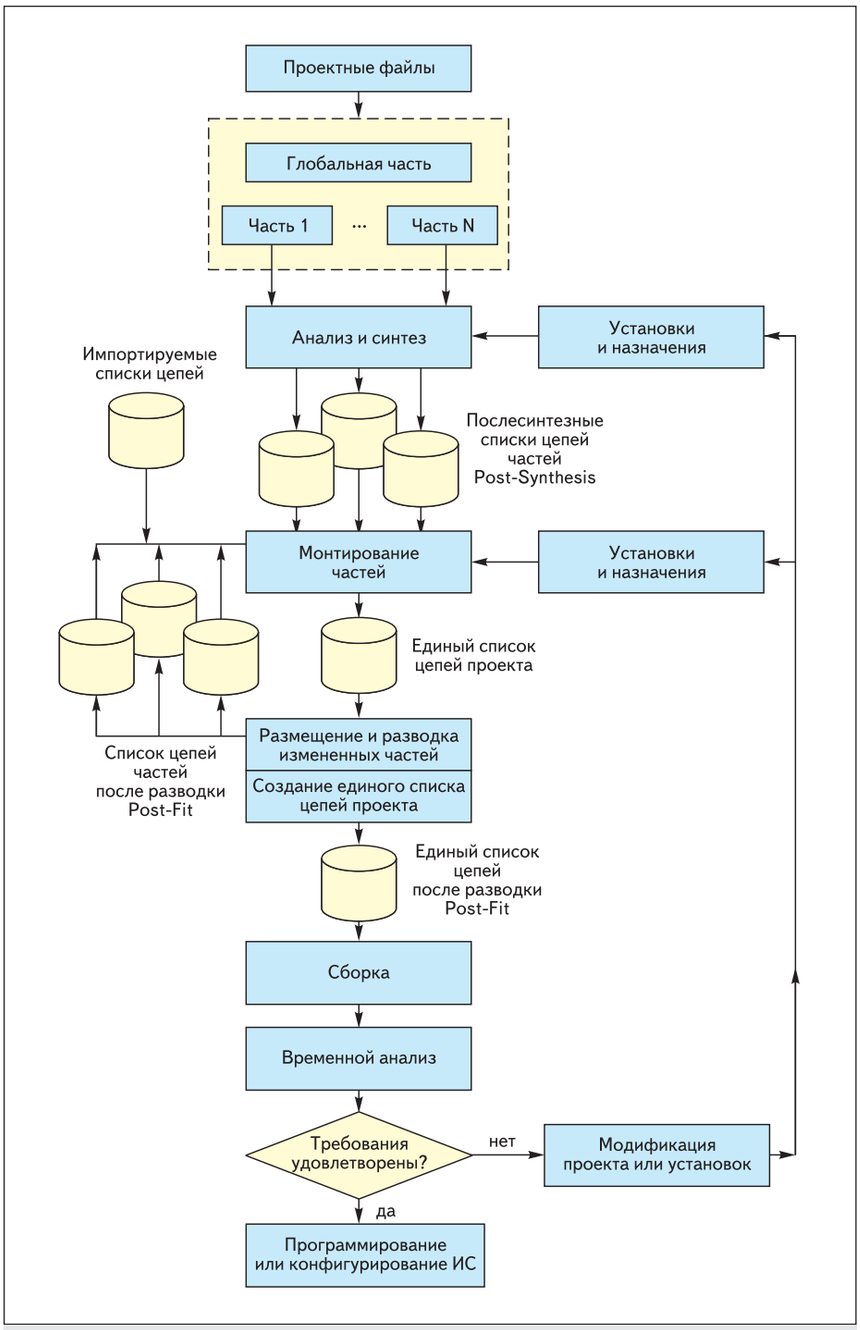

Рис. 2. Структурная схема инкрементального потока компиляции

водка фрагмента может осуществляться не только путем схемотехнических вариаций, но и путем изменений опции проектирования исключительно для отлаживаемого фрагмента.

Потоки проектирования «снизу-вверх» и «сверху-вниз»

Инкрементальная компиляция позволяет ускорить процедуру компиляции и дает возможность задать определенную последовательность процесса проектирования. При сложных разработках приходится ориентироваться на бригадные методы проектирования и использовать различные стратегии проектирования. Принципы инкрементальной компиляции позволяют проектировать как по схемам «сверху-вниз» и «снизу-вверх», так и в их произвольной комбинации.

Поток проектирования «сверху-вниз»

Стратегия проектирования «снизу-вверх» обычно применяется, когда стоит задача объединения результатов работы различных проектировщиков и поставщиков IP-ядер в единый поток компиляции. Эта стратегия традиционно поддерживается большинством САПР. Эту же стратегию использует один проектировщик при разработке модифицированного проекта, когда возникает проблема добавления новых блоков к ранее разработанным.

Проектный поток «снизу-вверх» на первом этапе допускает создание и оптимизацию каждого блока как отдельного проекта нижнего уровня. После разработки всех модулей, включая импортируемые блоки (IP) сторонних фирм, можно переходить к созданию имплементируемого (встраиваемого) проекта, который будет объединять готовые модули в проект верхнего уровня иерархии. Осуществляющий такое слияние проектировщик создает для себя «скелет» будущего проекта, отражающий иерархию ниже стоящих модулей. В новом (объединяющем) проекте вводятся глобальные установки для проекта целиком, назначаются тактовые сигналы для модулей и вводятся требуемые ограничения на связывающие сигналы.

Уже разработанные модули, которые были самостоятельными проектами, необходимо перевести с верхнего уровня иерархии на низкие уровни. Для каждой проектной части необходимо определить тип используемого списка цепей (Post-Synthesis netlist или Post-Fit netlist). Для того чтобы включить какой-либо технологический раздел в общий проект с неизменными свойствами, необходимо создать соответствующий LogicLock регион. В каждом технологическом разделе заданы список цепей и размеры региона. Дальнейшая реализация объединения происходит за счет механизмов экспорта и импорта. В этом случае руководитель проекта обязан организовать труд разработчиков низкоуровневых блоков таким образом, чтобы гарантировать правильное распределение ресурсов между отдельными разделами проекта.

Поток проектирования «сверху-вниз»

Необходимо отметить, что, если традиционно сохранности ранее полученных результатов компиляции можно было добиться только с использованием потока «снизу-вверх», то, благодаря инкрементальной компиляции, проектировщикам для достижения тех же целей стал доступен метод «сверху-вниз». При таком подходе отдельные части проекта, как и при стратегии «снизу-вверх», могут создавать разные разработчики. Однако компиляция и оптимизация проекта как единого целого на верхнем уровне иерархии осуществляется не после окончания проектирования всех отдельных блоков, а на самом первом шаге проектирования и производится только одним человеком. Как правило, за это действие отвечает руководитель проекта или ведущий разработчик.

Поскольку многие блоки еще не доведены до уровня реализации в ПЛИС, ведущий разработчик может создать для них пустые разделы. Для этих логических разделов требуется интерфейсная оболочка. Кроме того, необходимо зарезервировать размеры соответствующих им топологических разделов.

Для проекта в целом необходимо задать требуемые установки и указать интерфейсные сигналы интегральной схемы. Далее можно откомпилировать готовые разделы и проект целиком, после этого сохранить результаты трансляции и продолжить работу над новыми блоками.

Проектирование по рассматриваемой стратегии оказывается очень важным как минимум по двум причинам. Во-первых, поток «сверху-вниз», как правило, более прост в реализации, чем поток «снизу-вверх». Например, отпадает необходимость в экспорте и импорте проектов нижнего уровня. Во-вторых, при нисходящем подходе программному обеспечению САПР доступна информация о проекте в целом, что позволяет производить глобальную оптимизацию. Восходящий метод проектирования требует проведения дополнительной балансировки емкостных и временных ресурсов, поскольку при компиляции отдельных низкоуровневых разделов программное обеспечение не имеет ни малейшего представления об остальных, имеющихся в системе разделах.

Смешанный поток проектирования

Реальное проектирование сложных проектов обычно требует применения смешанной стратегии, объединяющей в себе достоинства схем «сверху-вниз» и «снизу-вверх». Как правило, начало проектирования соответствует стратегии «сверху-вниз», однако на последующих этапах некоторые разработчики пользуются преимуществами стратегии «снизу-вверх». Но в любой момент времени (в тех случаях, когда это необходимо) разработчик отдельного блока может выполнить компиляцию цельного проекта и определить влияние своего блока на свойства всего проекта.

Окончание следует

Литература

1. Quartus II Version 7.0 Handbook Volume 1: Verification (Section I. Design Flows). **w ww.altera.c om**
2. Описание САПР RTLvision PRO. **w ww.concept. de**
3. Компиляция в Quartus II. **ht tp://w ww.dsioffe.narod. ru**
4. Грушвицкий Р., Михайлов М. Проектирование в условиях временных ограничений: отладка проектов // Компоненты и технологии. 2007. № 6.
5. Грушвицкий Р., Михайлов М. Проектирование в условиях временных ограничений: отладка проектов // Компоненты и технологии. 2007. № 9.
6. Комолов Д. А., Мяльк Р. А., Зобенко А. А., Филиппов А. С. Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II. Краткое описание и самоучитель. М.: ИП РадиоСофт, 2002.