Окончание. Начало в № 1`2008

Методы и программные продукты для повышения производительности проектов на базе ПЛИС Xilinx

В предыдущих частях статьи были рассмотрены методы достижения максимальной производительности проектов на базе FPGA, основанные на «стратегических» мероприятиях — выборе правильного стиля кодирования, применении глобальных настроек САПР. В завершающей части показываются способы тонкой настройки проекта с помощью средств низкоуровневого редактирования топологии кристалла и задания индивидуальных временных ограничений отдельным цепям.

Илья ТАРАСОВ,
к. т. н.
tile@kc.ru

Способы оценки тактовой частоты средствами САПР

При определении тактовой частоты, на которой может работать проект, можно воспользоваться информацией, получаемой после различных этапов работы САПР ISE. Существуют три контрольные точки, которые позволяют определить производительность ПЛИС.

- Post-synthesis — самая ранняя оценка, производимая на основе логического анализа схемы и приблизительных предположений о ее последующей трассировке. В силу того, что информации о размещении компонентов на кристалле еще нет, точные данные существуют только для логических элементов. Задержка на трассировочных линиях оценивается на базе fanout (коэффициента разветвления по выходу) — цепи, которую необходимо подвести к большему числу входов, приписывается и бóльшая задержка. Итоговая частота может иметь отличия до 20% в любую сторону.
- Post-map — оценка, базирующаяся на отображении элементов схемы на ресурсы кристалла. Задержка на логических ячейках определяется точно, задержка на трассировочных линиях принимается равной 0. Для оценки реальной частоты на основе этой информации следует использовать «правило 60/40» — уже определенная задержка на логике принимается за 60% от общей задержки распространения сигнала. Количественные рамки являются рекомендательными (можно попробовать, к примеру, 80/20, однако у САПР могут возникнуть существенные затруднения при выполнении такого требования).
- Post-Place&Route — информация после этого этапа включает в себя точные сведения о задержках распространения сигналов не только по ячейкам ПЛИС, но и по реально выполненным трассировочным линиям (включая влияние температуры и напряжения питания). Это наиболее точная оценка, однако большое время, требуемое для выполнения размещения и трассировки, затрудняет ее постоянное использование на ранних этапах разработки проекта, когда желательно отслеживать эффект от произведенных изменений в коде как можно оперативнее. Тем не менее, окончательным свидетельством соответствия проекта заданным характеристикам является именно эта оценка.

Часто возникают ситуации, когда информация о допустимой тактовой частоте, предоставляемая САПР, оказывается заниженной и не соответствует реальным экспериментальным замерам. В данном случае речь идет не о повышении частоты сверх допустимых значений (overclocking), игнорируя требования производителя, а всего лишь о неполной информации о проекте, которую САПР может получить на основе анализа схемы и отдельных модулей. Иногда именно корректное и полное описание особых случаев позволяет получить от САПР корректную оценку тактовой частоты, которая оказывается существенно выше первоначальной. Для прояснения этого важного вопроса рассмотрим структуру временных ограничений более подробно.

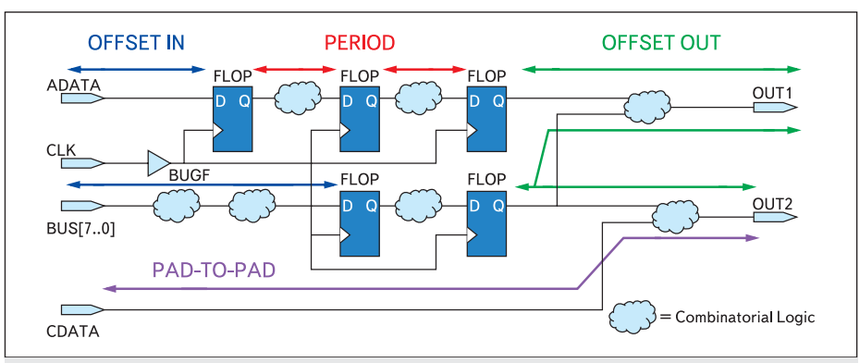

Рис. 1. Ограничения по распространению сигналов, применимые к различным узлам FPGA

Структура временных ограничений

В проекте на базе ПЛИС вводятся следующие основные ограничения по распространению сигналов (timing constraints), которые проверяются САПР (рис. 1). Все параметры задают максимальное время распространения сигналов по цепям каждого из классов. Превышение этого времени любой из цепей трактуется как ошибка.

Параметр PERIOD задает ограничение на максимальное время распространения сигнала от выхода одного синхронного элемен-

та до входа другого. Наиболее показательным примером является триггер (Flip-Flop), хотя блочная память и блоки DSP в синхронном режиме также будут подлежать данному ограничению. Таким образом, если задать значение PERIOD, то все цепи проекта смогут работать на частоте, соответствующей этому периоду. Часто оказывается, что задание этого единственного параметра позволяет обеспечить корректную работу всего проекта, поскольку при синхронном подходе к проектированию практически все цепи оказываются покрытыми параметром PERIOD.

Для задания данного параметра в файл проектных ограничений (с расширением ucf, User Constraints File) необходимо добавить строки вида:

```
NET "clk_i" TNM_NET = "clk_i";
TIMESPEC "TS_clk_i" = PERIOD "clk_i" 25 ns HIGH 50 %;
```

Также можно воспользоваться приложением PACE, позволяющим ввести проектные ограничения с помощью графического интерфейса.

Следующие параметры, OFFSET IN и OFFSET OUT, подобны друг другу. Первый из них задает время распространения сигналов от входа ПЛИС до входов синхронных компонентов (учитывая и время t_{setup}, необходимое для надежного захвата триггером нового состояния). Второй задает максимальное время от записи сигналов в триггеры до их появления непосредственно на выходах ПЛИС. Как показано на рис. 1, в этом случае принимаются во внимание и блоки комбинаторной логики, поэтому очень важно конвейеризовать обработку, записывая сигнал в триггер как можно раньше и подавая его на выход непосредственно с триггеров. Для этого полезно использовать триггеры блоков ввода/вывода, которые идеально подходят для этой цели и обеспечивают прекрасные характеристики обоих параметров OFFSET.

Параметр PAD-TO-PAD соответствует цепям, проходящим сквозь кристалл, без промежуточной фиксации в триггерах. Очевидно, что задержка в таких цепях будет максимальной, поскольку она включает в себя и элементы, обусловливающие время OFFSET IN, и элементы, обусловливающие OFFSET OUT. Кроме того, дополнительную задержку внесет комбинаторная логика внутри ПЛИС и трассировочные линии, особенно если сигнал передается между выводами, далеко отстоящими друг от друга.

Как уже упоминалось, минимальным действием со стороны разработчика является задание параметра PERIOD, который сразу обусловит частоту работы внутренних синхронных ресурсов. Как правило, параметры OFFSET играют при этом меньшую роль, особенно если разрешено использование встроенных триггеров блоков ввода/вывода. Тем не менее, при необходимости можно ввести и индивидуальные ограничения на какие-либо цепи. Примером таких индивидуальных ограничений может являться явное задание параметра Multi-Cycle Factor.

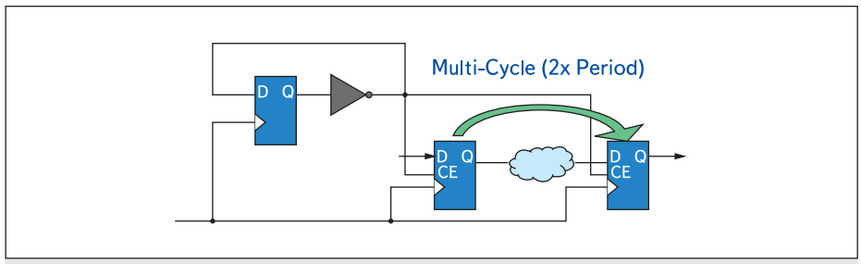

Рис. 2. Иллюстрация к понятию Multi-Cycle Path

Смысл этого уточнения заключается в том, что не все триггеры проекта, тактируемые сигналом некоей частоты, переключаются именно с этой частотой. Такая цепь, состояние которой гарантированно меняется на каждый такт, обозначается Multi-Cycle Path. В качестве примера рассмотрим рис. 2.

На рис. 2 показаны два триггера, тактируемые некоторой частотой. Однако на вход «разрешение счета» активный уровень подается только каждый второй такт. Следовательно, состояние первого триггера будет изменяться с частотой f/2, и максимальная задержка распространения сигнала до второго триггера может быть в два раза больше, чем период тактового сигнала. Цепь, показанная «облачком», и является Multi-Cycle цепью с Multi-Cycle Factor = 2. Смысл указания этого параметра в том, что при временном анализе САПР будет сравнивать задержку на этой цепи не с периодом тактового сигнала, а с периодом, умноженным на Multi-Cycle Factor (в данном случае 2). Если цепь является сложной, задержка в ней вполне может оказаться больше периода тактового сигнала, и САПР сообщит о невозможности работы проекта на указанной частоте. Если же есть явное указание, что особенности схемы не требуют распространения сигнала конкретно по данной цепи каждым тактом, это не приведет к возникновению ошибки.

Кроме Multi-Cycle Factor, можно отмечать цепи, задержки на которых следует игнорировать (параметр TIG, timing ignore). Например, это может быть цепь, подключенная к внешней кнопке, — очевидно, что задержка между нажатием кнопки человеком и реакцией ПЛИС, вносимая микросхемой, неизмеримо больше реакции человека.

В целом может оказаться, что одна и та же цепь подлежит одновременно многим ограничениям — например, общее ограничение PERIOD, частное ограничение на шину, еще более строгое ограничение на один из разрядов этой шины (который оказался наиболее медленным, поэтому требования по его трассировке оказались ужесточены). В данном случае приоритет ограничений повышается именно в том порядке, в котором они были перечислены — итоговым требованием окажется индивидуальное ограничение времени распространения по данной линии. Сложная структура этих ограничений позволяет гибко регулировать работу САПР по трассировке проекта, проводя сначала наиболее «проблемные» цепи, в дальнейшем дополняя их теми, трассировка которых проще.

При изучении процесса введения конструкторских ограничений можно руководствоваться схематичными рекомендациями:

- проанализировать временные задержки в цепях проекта без ограничений;
- в первую очередь устанавливать параметр PERIOD;
- не вводить индивидуальные ограничения на цепи сверх необходимости.

Настройка режимов входных буферов в ПЛИС Virtex-4/5

В high-end ПЛИС Virtex-4/5 есть возможность тонкой подстройки фазы тактового сигнала (в блоках DCM) и задержек на входных буферах (IDELAY). Это связано с необходимостью обеспечивать правильные временные диаграммы для синхронных цифровых элементов (в первую очередь триггеров). Для надежной записи данных нужно, чтобы логический уровень на входе D был выставлен за время как минимум t_{setup} до фронта тактового сигнала Clk. Однако от появления логических уровней на выводах ПЛИС до их установления на соответствующих входах триггера проходит некоторое время, зависящее от того, какие именно трассировочные линии были для этого использованы. В итоге временные диаграммы могут выглядеть так, как показано на рис. 3.

Оба сигнала, тактовый и данных, при прохождении до триггера получают некоторую задержку. Технологический разброс, зависимость от температуры и прочие факторы обусловливают колебания величины этой задержки в пределах как серии изделий, так и одного и того же экземпляра ПЛИС. В конечном итоге, чтобы быть уверенными, что нужный логический уровень на входе данных действительно пришел на t_{setup} раньше, следует взять максимальную задержку распространения данных $T_{Data(Max)}$ и минимальную — тактового сигнала ($T_{Clk(Min)}$). Получаемый в этом случае запас по времени соот-

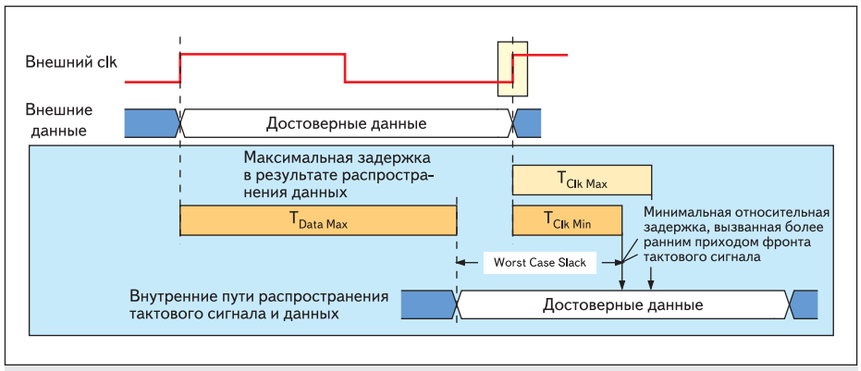

Рис. 3. Временные диаграммы сигналов, проходящих через входные буферы FPGA

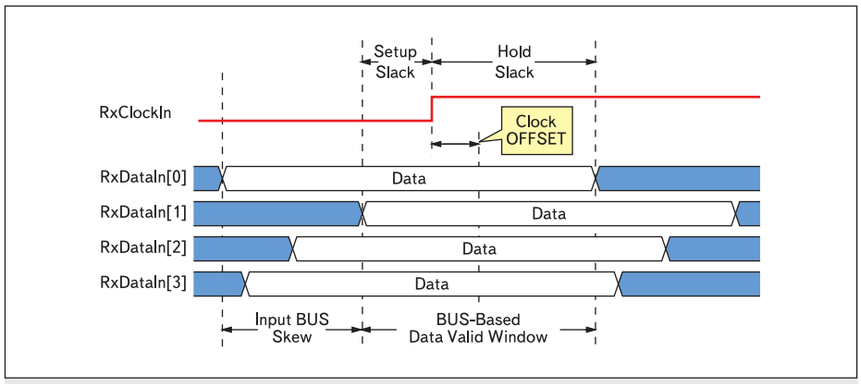

Рис. 4. Пример неравномерных задержек в отдельных разрядах шины

ветствует наихудшей ситуации (Worst Case Slack). Очевидно, что этот запас можно делать настолько маленьким, чтобы он только-только соответствовал t_{setup}, указанному в документации на ПЛИС. Для этого можно как сдвинуть фазу тактового сигнала путем соответствующей настройки DCM, так и изменить задержку на линии данных настройкой элемента задержки IDELAY. В ПЛИС Virtex разрешающая способность этих настроек весьма хороша и составляет 1/256 от периода тактового сигнала, или представляет собой величину порядка десятков пикосекунд для линии IDELAY (с максимум 64 элементами в линии задержки). Кроме того, доступна динамическая реконфигурация задержек, например, при изменении температуры кристалла, измеряемой соответствующим датчиком. Именно эти возможности и позволяют семейству Virtex-5 работать с памятью DDR-667 и DDR-800, а меньшая функциональность настроек у Spartan-3 и отсутствие режима динамической реконфигурации DCM не позволяют подняться им выше DDR-333. В качестве примера можно привести рис. 4, где проиллюстрирована проблема организации многоразрядных синхронных шин. Неравномерность задержек в отдельных разрядах данных может привести к тому, что надежное выдерживание времени установки и времени удержания может оказаться невозможным. Для этого необходимо устанавливать индивидуальные задержки во входных цепях, выравнивающие суммарное время распространения разрядов до входов отдельных регистров, в то время как центрирование фронта тактового сигнала относительно интервала, в котором данные достоверны, можно выполнить подстройкой фазы Clk.

Можно отметить, что достижение высокой тактовой частоты при обмене с внешними устройствами, особенно по параллельным интерфейсам, требует серьезного анализа и тщательного учета различных факторов, не ограничиваясь приведенным выше. САПР ISE позволяет учесть также неопределенность положения фронта, джиттер, скорость нарастания фронтов и т. д. Подробнее эта информация излагается на сайте Xilinx, а также в учебном курсе Advanced FPGA Implemen-tation.

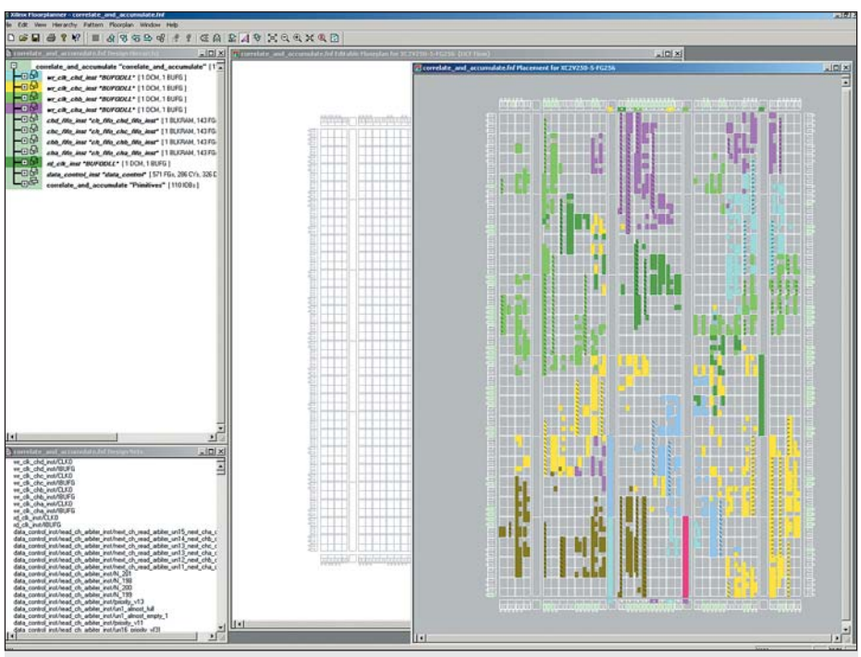

Рис. 5. Анализ и редактирование размещения модулей проекта в программе FloorPlanner

Управление размещением компонентов в программе Floorplanner

Если меры, принятые на «стратегическом» уровне, все еще не позволили получить требуемую тактовую частоту, можно перейти к низкоуровневому планированию топологии проекта. Для этого служит приложение FloorPlanner, внешний вид которого показан на рис. 5.

Как видно на рис. 5, приложение представляет «вид сверху» на кристалл ПЛИС, с выделением цветом расположения отдельных модулей проекта. Цвет соответствует обозначениям отдельных компонентов, перечисленных в панели слева от основного окна. Критерием автоматического разбиения на такие компоненты является выделение разработчиком отдельных модулей и наличие цепочек ускоренного переноса (так что счетчики, сумматоры, аккумуляторы, для которых критично относительное расположение отдельных ячеек, оказываются выделен-

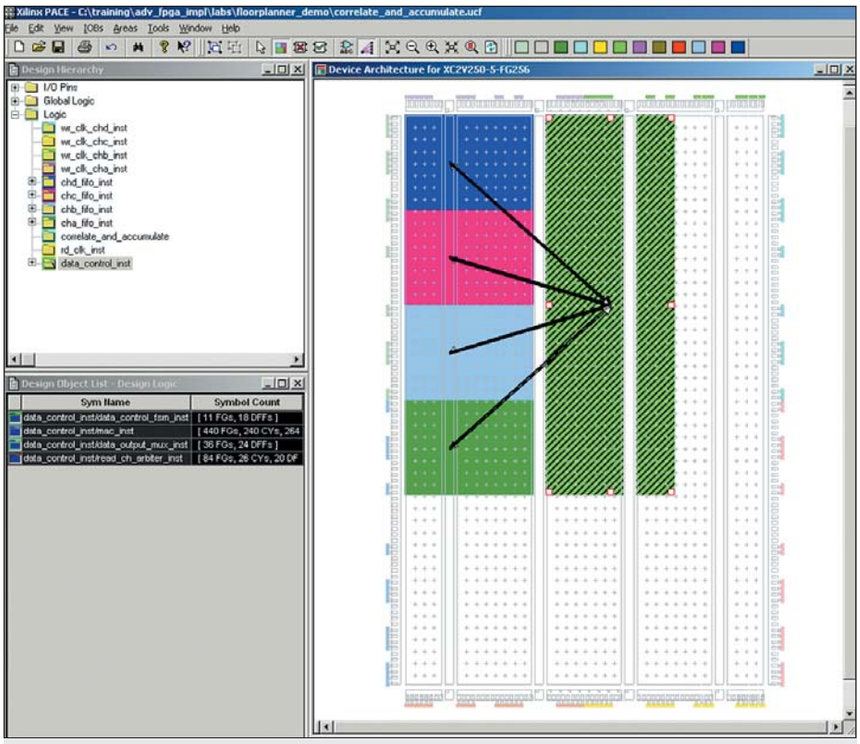

Рис. 6. Введение проектных ограничений в виде указания расположения модулей с помощью приложения FloorPlanner

ными в отдельные элементы). На рис. 5 узлы, использующие цепи ускоренного переноса, легко идентифицируются в основном окне — они представляют собой узкие вертикальные столбики ячеек (их можно спутать с блоками встроенной памяти, которые имеют подобный вид, но располагаются в отдельных колонках). Показанная на рис. 5 информация образуется в ISE после выполнения этапа Place&Route, то есть когда все элементы схемы привязаны к конкретным ячейкам. В этот момент можно отредактировать положение компонентов с целью разместить критические цепи как можно более компактно. Очевидно, что эта работа весьма трудоемка, а неосторожная коррекция схемы может *ухудшить* характеристики проекта вместо улучшения. Именно поэтому работу с FloorPlanner рекомендуется проводить не только когда все предварительные мероприятия уже проведены (использование аппаратных модулей, синхронный стиль описания, настройка средств синтеза), но и при наличии у разработчика достаточного опыта работы с ПЛИС (Xilinx рекомендует не менее двух лет), глубокого понимания архитектуры выбранного семейства и возможностей САПР.

Также с помощью FloorPlanner можно выполнить схематичное обозначение расположения отдельных модулей, которое гораздо менее трудоемко, но, тем не менее, способно весьма позитивно повлиять на финальные характеристики проекта. Пример показан на рис. 6.

На рис. 6 показан демонстрационный проект, состоящий из четырех независимых FIFO и мультиплексора, выполняющего постобработку одного из выбранных каналов. Реализация этого проекта «по умолчанию» приводит к не вполне удовлетворительным результатам, а FloorPlanner позволяет наблюдать, что ресурсы, относящиеся к отдельным каналам, перемешаны, из-за чего длина связей между ресурсами в пределах одного канала чрезмерно возрастает. Однако если после выполнения процесса Map ввести проектные ограничения, указав области, в которых должны находиться отдельные каналы и объединяющее устройство постобработки, как показано на рис. 6, тактовая частота возрастает почти на 50%. Черные линии на рис. 6 не являются добавленной графикой, а рисуются программой для визуализации связей между блоками, так что можно наглядно убедиться, что между четырьмя отдельными каналами нет связей и можно свободно разнести их по кристаллу.

Редактирование топологии кристалла

Если после проведения рассмотренных выше мероприятий разработчик все же недоволен получаемыми результатами, и проблема, на его взгляд, кроется в неудачном проведении трассировочных линий, можно провести дальнейшее углубление в проект, использовав приложение FPGA Editor (рис. 7).

Этот инструмент позволяет получить доступ практически к всем ресурсам ПЛИС на наиболее низком уровне, включая трассировочные ресурсы, внутренние ресурсы логических ячеек, режимы работы отдельных устройств, состояния таблиц истинности и т. д. Фактически FPGA Editor мог бы быть использован для «ручного» создания схемы, однако трудоемкость при этом оказывается неприемлемо высокой. Более приемлемый путь использования низкоуровневого представления топологии — наблюдение трассировочных линий и выявление критичных участков.

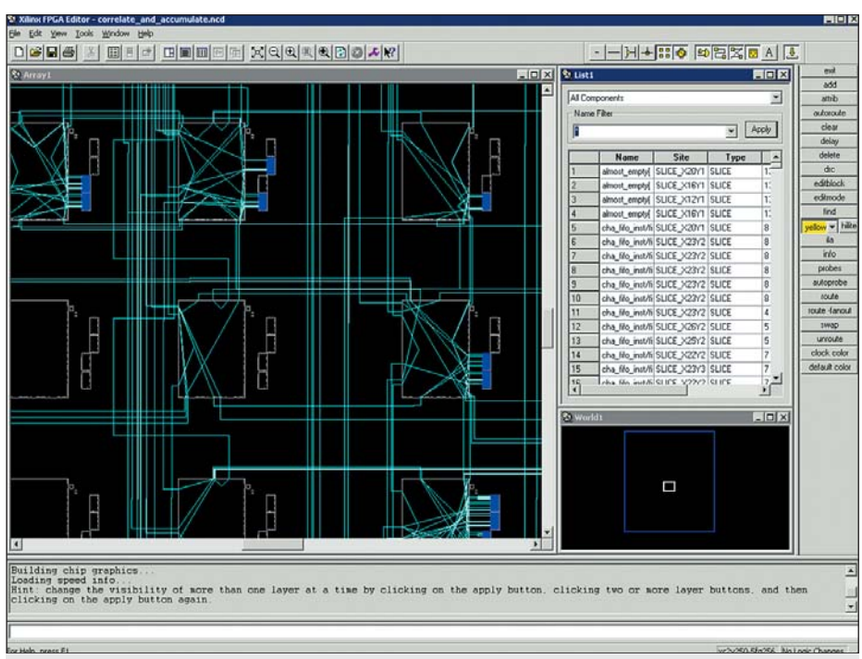

Рис. 7. Окно приложения FPGA Editor

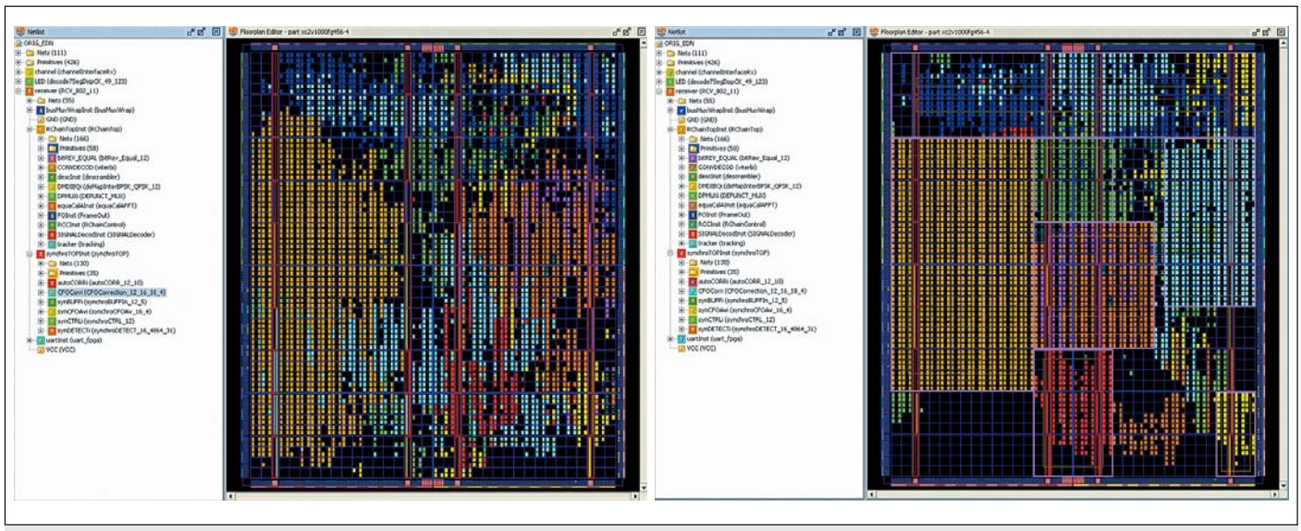

Рис. 8. Пример использования PlanAhead для оптимизации размещения модулей (до и после оптимизации)

Можно заметить, что FloorPlanner позволял управлять только процессом размещения логических ячеек, но не проведения связей между ними. Для того чтобы оценить масштабы работ, которые необходимо проводить для работы с FPGA на этом уровне, достаточно обратить внимание на правый нижний угол окна программы на рис. 7, где синим прямоугольником показаны границы кристалла, а белым — видимая в основном окне область. Иными словами, речь идет о тысячах ячеек, доступных для прямого редактирования, причем в каждой ячейке находятся десятки регулируемых элементов, а вокруг нее — множество сегментов трассировочных линий. Ввиду этого работа в FPGA Editor является весьма специфичной и в подавляющем большинстве проектов общего назначения не требуется.

САПР PlanAhead — мощное средство управления размещением модулей

Чрезмерно высокая трудоемкость работы в программе FloorPlanner с большими FPGA, в сочетании с возможностью получения хороших результатов, способствовала появлению продукта PlanAhead. Это отдельно приобретаемый программный продукт, основным назначением которого является обеспечение более глубокого, чем у FloorPlanner, уровня анализа проекта и управления размещением ресурсов в достаточно больших FPGA (рекомендация Xilinx — начиная с XC4VLX15). При работе с FloorPlanner можно быстро обнаружить, что его можно использовать двумя весьма разными способами: введением проектных ограничений на ранних стадиях работы с проектом (это делается относительно просто, а для начала такой деятельности достаточно представлять общую структуру проекта, примерный объем модулей и интенсивность связей между ними) и низкоуровневым редактированием размещения компонентов на уровне логических ячеек. Напрашивается некий промежуточный уровень — гибкое объединение ячеек и отдельных узлов в более крупные блоки, разбиение чрезмерно крупных блоков на более мелкие, работа со сценариями «а что, если…». Именно этот уровень и предлагает продукт PlanAhead — управление иерархией модулей проекта уже в процессе размещения и трассировки. Для этого вводится понятие Physical Block (Pblock), такие блоки могут динамически создаваться и объединяться в процессе работы над проектом. Иными словами, разработчик в процессе анализа размещения проекта и временных задержек может объединять и фиксировать блоки, которые в процессе разработки оказались успешными. Более того, полученные блоки с учетом их размещения могут впоследствии применяться в других проектах, выполняемых на том же семействе FPGA (технология повторного использования IP-ядер). Кроме того, если некий Pblock оказался слишком громоздким и вследствие этого сложным в трассировке, его можно разбить на два или более подблоков, с тем чтобы оптимизировать их по отдельности.

Отмечается, что в среднем повышение производительности при использовании PlanAhead составляет 10–15%, хотя в отдельных случаях может достигать 30%. Принципиальным является тот факт, что при этом может оказаться возможным использование ПЛИС меньшей градации скорости, что для больших FPGA означает очень существенную экономию средств. Если рассмотреть технические характеристики FPGA с разной градацией скорости, можно заметить, что повышение тактовой частоты с помощью PlanAhead как раз и позволяет использовать ПЛИС минимальной градации в том же режиме, что и более дорогой кристалл того же наименования, но следующей градации скорости. Разумеется, повышение производительности достигается исключительно путем более рационального размещения модулей относительно друг друга, а не какого-либо ускорения работы самих физических ресурсов. Однако анализ цен на большие кристаллы позволяет говорить, что даже при относительно небольшой партии выпускаемых изделий экономия от перехода к менее скоростным кристаллам может оказаться значительно больше стоимости PlanAhead и сопутствующих расходов на разработку.

На рис. 8 показан пример использования PlanAhead для оптимизации размещения модулей. Подробная информация о продукте помещена на сайте Xilinx **www.xilinx.com/planahead**.

Заключение

Развитие FPGA большого объема и усложнение их архитектуры привело к тому, что автоматические средства трассировки кристаллов далеко не всегда способны дать приемлемое решение с установками по умолчанию. Ввиду этого возрастает роль квалифицированных специалистов, понимающих особенности архитектуры FPGA, владеющих современными и эффективными методами проектирования и программными инструментами для оптимизации размещения модулей на кристалле ПЛИС. Наличие различных градаций скорости в рамках одного семейства FPGA делает возможным получение конкретной экономической выгоды от применения оптимизации.

Подробная информация о рассмотренных методиках проектирования содержится в учебных материалах фирмы Xilinx, о которых можно узнать по адресам: **http://plis.ru/page.php?id=57** и **http://plis.ru/page.php?id=58**.