ОШИБОК НЕ БУДЕТ
УМНЫЕ ТЕХНОЛОГИИ ЛОГИЧЕСКОГО СИНТЕЗА И ОТЛАДКИ

А.Сэттон, старший менеджер по маркетингу, Synopsys Inc

Причина того, что проект для ПЛИС не может быть синтезирован или работает не так, как ожидалось, может быть не очевидна. Среди тысячи файлов RTL-кода и ограничений (constraints) (которые, возможно, созданы другими инженерами) обнаружить ошибку крайне сложно. Учитывая длительность итераций сложных проектов на ПЛИС, разработчики должны иметь возможность находить и устранять ошибки на ранних стадиях проектирования и уделять больше внимания верификации проекта.

Очень важно совершенствовать методы нахождения ошибок для каждого конкретного случая – уметь определять, какие части схемы работают неправильно, найти их соответствия в исходном RTL-коде и внести в него необходимые исправления. Можно сэкономить немало времени, выполнив простейшую проверку проекта на синхронность (timing check), проконтролировав ограничения на уровне интерфейсов модулей до проведения многочасовых процедур синтеза, размещения и трассировки (P & R).

Для решения этих задач разработчики могут использовать инструменты САПР Synopsys Synplify Premier и Synplify Pro (для синтеза в базис ПЛИС) и Identify RTL Debugger (для отладки RTL-кодов). Эти инструменты позволят быстро локализовать ошибки и существенно сократить продолжительность и число итераций, необходимых для создания работоспособных проектов на ПЛИС.

ЛОКАЛИЗАЦИЯ ИСТОЧНИКА ОШИБОК

Если функциональные ошибки обнаруживаются, когда ПЛИС уже установлена на печатной плате, найти их причины бывает достаточно сложно. В этом случае для отладки проекта понадобятся дополнительные схемотехнические решения. Чтобы анализировать и изменять данные, полученные при работе схемы, необходимо иметь возможность сохранения состояния определенных внутренних узлов схемы в ПЛИС. Рассмотрим способы обнаружения ошибок в проекте с помощью программного обеспечения Identify RTL Debugger.

Алгоритм поиска ошибок с использованием RTL-отладчика состоит из четырех шагов. Для отслеживания нужных сигналов задаются контрольные точки. Затем они "отображаются" в исходном RTL-коде. Это позволяет выявить причину ошибок в коде или в заданных требованиях и ограничениях.

Шаг 1. Назначение контрольных точек. В RTL-коде указываются сигналы и условия, которые нужно отследить. Можно установить точки наблюдения (для сигналов или узлов) либо точки останова (для условных и безусловных ветвлений, например, в конструкциях типа if, then и case).

Шаг 2. Сборка проекта с контрольными точками. Проект синтезируется в базис ПЛИС вместе с заданными контрольными точками

и интеллектуальным внутрисхемным эмулятором (IICE), который считывает и экспортирует отладочные данные в соответствии с созданными контрольными точками.

Шаг 3. Анализ и отладка. После окончания синтеза, размещения и трассировки проект загружается в ПЛИС и запускается. Встроенный эмулятор начинает накапливать данные с контрольных точек. Выборка данных может быть записана в файл .vcd и отображена в визуализаторе RTL-кода или в инструменте моделирования.

Шаг 4. Внесение исправлений в проект. Как только ошибка будет обнаружена, ее можно исправить непосредственно в RTL-коде или файле ограничений (в зависимости от того, где произошла ошибка), используя иерархический или не иерархических подход.

ВИЗУАЛЬНЫЙ КОНТРОЛЬ ПАРАМЕТРОВ И ФУНКЦИОНАЛЬНЫХ ОШИБОК

В инструментах Synopsys для проектирования и отладки ПЛИС имеется возможность представить RTL-код и список соединений (netlist) в графическом виде. Например, инструмент визуализации схемы с интерактивной отладкой отображает RTL-код и список соединений схемы и позволяет соотнести результаты проверки на синхронность и данные, собранные с контрольных точек в файл .vcd, с исходным RTL-кодом. Схематическое представление проекта в виде технологически независимых компонентов (сумматоры, регистры, мультиплексоры и управляющие конечные автоматы) под названием RTL View становится доступным при компиляции RTL-кода. Из любой точки этой схемы можно перейти в соответствующую точку исходного кода и изменить его, если реализация или поведение схемы не соответствует задуманному, либо в редактор ограничений, в котором можно легко обновить или задать новые ограничения (рис.1).

Чтобы обнаружить причину неправильного поведения схемы в исходном RTL-коде, можно воспользоваться функцией RTL debugger, которая отображает данные, собранные в контрольных точках, на соответствующих узлах и элементах схематического представления RTL-кода.

Визуализаторы также позволяют отобразить так называемое технологическое представление уровня списка цепей (Technology View) – фактическую реализацию проекта в базисе ПЛИС после синтеза. В визуализаторе HDL Analyst для этого используются примитивы Xilinx – таблицы (lookup tables), регистры и элементы цифровой обработки сигналов (DSP). От любой цепи или сигнала в этом представлении можно перейти к исходному RTL-коду, а также к отчетам о быстродействии как после синтеза, так и после размещения и трассировки, что позволяет оценить и повысить производительность.

ОТЛАДКА БОЛЬШИХ ПРОЕКТОВ

Создание контрольных точек для всех без исключения сигналов в большом проекте невозможно, так как суммарный объем данных, генерируемых в этих точках, будет астрономическим, а площадь, занимаемая необходимой для сбора этих данных дополнительной логикой, – слишком велика. Общая проблема методологии со встроенной отладочной логикой заключается в том, что может быть трудно, а иногда и невозможно, заранее предсказать, на каких сигналах и узлах необходимо установить контрольные точки.

Некоторые программные отладчики позволяют частично решить эту проблему. Мультиплексируя группы контрольных точек, разработчики могут выборочно отслеживать сигналы и переключаться между ними, используя общий внутрисхемный эмулятор IICE. При таком подходе есть возможность наблюдать больше сигналов и контрольных точек без увеличения объема хранимых данных. Кроме того, можно переключаться между группами интересующих вас сигналов "на лету", без необходимости тратить время на повторное задание контрольных точек и синтез проекта.

К сожалению, внутрисхемный эмулятор использует аппаратные ресурсы ПЛИС, в том числе встроенные блоки оперативной памяти (RAM). Уменьшить количество занятых эмулятором внутренних блоков памяти можно, задействовав для накопления данных с контрольных

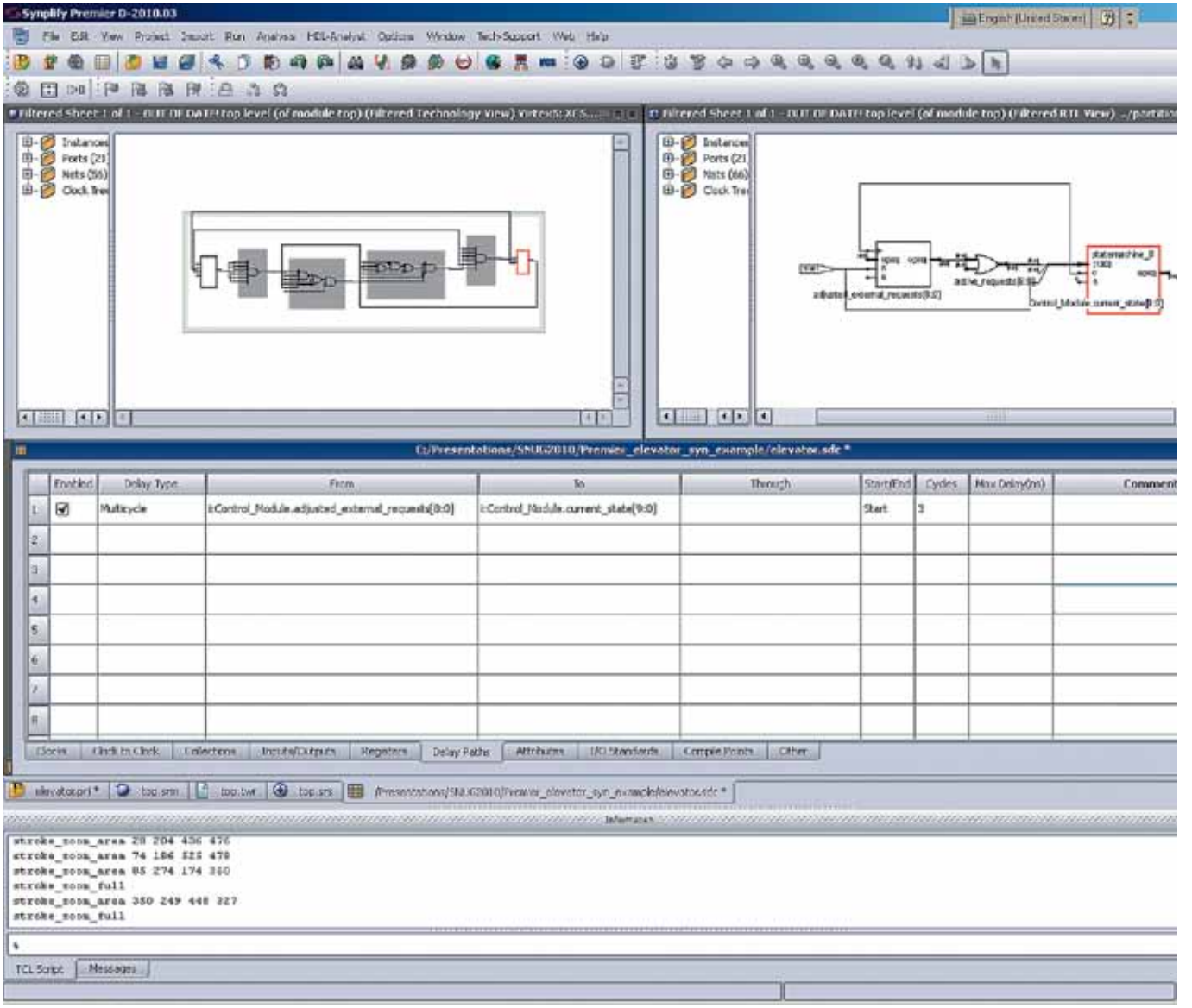

Рис.1. Графическое представление RTL-кода

точек внешнюю память (SRAM). Этот подход также позволяет увеличить глубину данных с контрольных точек.

МОЙ ПРОЕКТ НЕ СИНТЕЗИРУЕТСЯ!

Из-за некоторых ошибок не могут завершиться этапы синтеза, размещения и трассировки проекта. Если количество файлов исходного RTL-кода и ограничений велико, поиск ошибок и неверных ограничений, препятствующих выполнению этих этапов, может занять несколько недель. При создании прототипов на ПЛИС нужно проверить исходные файлы на пригодность к синтезу в базис ПЛИС – например, на необходимость преобразования схем управления тактированием.

Некоторые схемы управления тактированием, используемые при проектировании СБИС, не существуют в архитектуре ПЛИС. Они могут стать причиной нерационального использования ресурсов ПЛИС. Один из эффективных способов решения этой проблемы – возможность автоматической конвертации этих схем инструментами синтеза ПЛИС. Например, управляемый внутренней логикой или сгенерированный (вторичный) тактовый сигнал преобразуется так, что будет подаваться на вход разрешения (enable) секвенциального элемента, а не на вход тактового сигнала (clock). Это позволит подключать элементы секвенциальной логики непосредственно к источнику тактового сигнала, устранив проблему рассинхронизации фронтов

```
[#### START OF CLOCK OPTIMIZATION REPORT #####]

9 instances converted, 16 sequential instances remain driven by non-clean clocks

================================================== Gated/Generated Clock Combined Report =========+++==================
Clock Tree ID     Driving Element     Drive Element Type     Fanout     Sample Instance          Explanation
-------------------------------------------------------------------------------------------------------------------------
ClockId0001       comb1_u0.golk       LUT4                   8          ff_reg1_u0.out1[2]       Multiple clocks on instance
ClockId0002       comb2_u0.golk       LUT2                   8          ff_reg1_u3.out1[1]       No clock input found
=========================================================================================================================

[##### END OF CLOCK OPTIMIZATION REPORT ######]
```

Рис.2. Отчет об ошибках

тактовых сигналов, сократив количество источников тактирования схемы и сэкономив в итоге ресурсы ПЛИС.

Для того чтобы включить функцию автоматического преобразования управляемых тактовых сигналов в инструменте Synplify Premier, необходимо выбрать в меню Projects пункт Implementation Options и во вкладке GCC & Prototyping включить операцию Clock Conversion. Либо же выполнить в командной строке интерпретатора TCL-команду

```
-set _ option -fix _ gated _ and _ generated _ clocks 1
```

Эта команда разрешает автоматическое преобразование управляемых логикой тактовых сигналов или сгенерированных тактовых сигналов, в то время как команда

```
-set _ option -conv _ mux _ xor _ gated _ clocks 1
```

преобразовывает тактовые сигналы, используя ресурсы мультиплексоров или вентилей "ИЛИ", входящих в состав цепей тактовых сигналов для проектов на платформах HAPS.

Полный набор ограничений для тактовых сигналов подразумевает, что первичные тактовые сигналы определены везде, где это необходимо, также определены и все отношения между вторичными тактовыми сигналами. Преобразование тактовых сигналов может выполняться неправильно или завершиться ошибкой, если:

- отсутствуют какие-либо ограничения;
- отсутствуют тактовые сигналы, которые подаются на последовательный элемент;
- неверно установлены ограничения, из-за которых тактовый сигнал оказался отделенным от его истинного источника, например, из-за наличия "черного ящика" между источником тактового сигнала и последовательными элементами.

Во многих случаях невозможность преобразования тактовых сигналов связана с неполным или ошибочным набором ограничений. Например, в схеме может присутствовать комбинаторная петля, которая должна быть разорвана соответствующими ограничениями до преобразования. Отчет, создаваемый в процессе синтеза, содержит список преобразованных тактовых сигналов, их имена, типы, группы и соответствующие ограничения. В отдельном списке находятся имена сигналов, которые не удалось преобразовать, и сообщение об ошибке с объяснением причины ее возникновения (рис.2).

Если, в проекте присутствуют блоки типа "черный ящик", есть возможность "помочь" автоматическому преобразованию тактовых сигналов с помощью специальных директив в RTL-коде, понятных инструменту синтеза. Например, с помощью директивы syn_gatedclk_clock_en можно указать имя входа разрешения у блока "черного ящика", а с помощью директивы syn_gatedclk_clock_en_polarity – полярность тактового сигнала.

Каждый преобразованный объект и его тактовый сигнал получают свойства (properties), по которым они могут быть найдены в базе данных проекта. Информация о них выводится в соответствующий отчет, который создают команды группы Find интерпретатора TCL.

НЕСООТВЕТСТВИЕ ПАРАМЕТРОВ ПОРТОВ

Проект может включать в себя файлы исходного RTL-кода, написанные сторонними разработчиками. При использовании в проекте IP-ядер либо готовых блоков весьма часто возникают ошибки спецификации "несоответствие портов" (port mismatch), которые иногда весьма сложно обнаружить и устранить. Особенно это касается смешанных проектов, где одновременно применяется несколько языков описания аппаратуры. Например, в блоке верхнего уровня, написанном на VHDL, имеется компонент sub, написанный на Verilog. Verilog-модуль имеет порты разрядностью три бита, а верхний уровень подразумевает подсоединение четырехбитных портов. В этом случае, к примеру, Synplify Pro/Premier, немедленно обнаружит несоответствие разрядности шин и выдаст предупреждающее сообщение в виде гиперссылки на соответствующую статью руководства пользователя:

```
Interface Mismatch between View work.sub.syn _
black _ box and View work.sub.verilog
Details:
========
The following bit ports in the Source View work.sub.
syn _ black _ box do NOT exist in the Target View
work.sub.verilog
=========================================
    Bit Port in1[4]
    Bit Port in2[4]
    Bit Port dout[4]
```

Как можно узнать оригинальное имя модуля с неверным определением интерфейса, если проект имеет разветвленную многоуровневую иерархию? Для этого инструмент помечает все экземпляры модулей при помощи специального атрибута orig_inst_of. Имя родительского модуля, хранящееся в этом атрибуте, помогает легко восстановить взаимосвязи в проекте. Например, если экземпляр модуля sub под названием sub_3s вызывает ошибку несоответствия портов, можно узнать имя родительского модуля, используя команду на языке TCL

```
get _ prop -prop orig _ inst _ of {v:sub _ 3s}
```

которая возвращает значение sub.

ОТЛАДКА И ОПТИМИЗАЦИЯ ОГРАНИЧЕНИЙ

Задание разумных и правильных ограничений влияет не только на качество получаемых результатов синтеза, но и зачастую на правильность функционирования будущего изделия. Ограничения, как правило, должны включать в себя:

- определение тактовых сигналов и выделение их групп;
- объявление асинхронных тактовых сигналов;
- объявление ложных (false path) и многотактовых путей (multicycle path).

Общепринятой практикой и хорошим стилем работы считается обязательная проверка заданных ограничений до проведения синтеза. Инструмент, проверяющий ограничения на синтаксические ошибки, ошибки имен и возможность применения заданных ограничений к текущему проекту, позволит на раннем этапе предупредить о возможных проблемах. Это может быть, например, отчет о применении ограничений после раскрытия символов-шаблонов (?, * и т.д.) либо о взаимодействии пересекающихся тактовых сигналов после их задания. Кроме этого, может быть выдан список ограничений, проигнорированных, например, из-за отсутствующих объектов, на которые они ссылаются.

Для запуска анализа ограничений и создания отчета в инструментах Synplify Pro/Premier можно использовать как графический интерфейс (меню Run–Constraint check), так и командную строку интерпретатора TCL (в интерактивном либо пакетном режимах):

```
project -run constraint _ check
```

Отчет с именем projectName_cck.rpt сохраняется на диске и выводится на экран.

Кроме этого, для исключения возникновения метастабильности при пересечении двух асинхронных тактовых доменов следует запускать анализатор асинхронных тактовых сигналов, выдающий пользователю отчет о путях, начинающихся в одном домене и заканчивающихся в другом (пересечение доменов).

Для генерации отчета в режиме пользовательского интерфейса нужно в меню Analysis–Timing Analyst установить флажок Generate Asynchronous

Clock Report. С помощью интерпретатора TCL это выполняется командой

```
set _ option -reporting _ async _ clock
```

Отчет сохраняется на диск под именем projectName_async_clk.rpt.csv .

Хороший стиль проектирования подразумевает проверку задаваемых ограничений на полноту и разумность для текущего проекта без переограничений. Переограниченные, т.е. принципиально не способные работать в соответствии с заданными ограничениями проекты будут долго синтезироваться, а результат синтеза, скорее всего, будет содержать нарушения заданных ограничений.

Необходимо обратить внимание, что часто цепи-нарушители по сути ложны, т.е. не используются во время работы проекта. Зачастую пользователь имеет представление о том, что некоторые цепи в проекте ложные или многотактовые – их следует помечать при помощи соответствующих ограничений. Это относится и к цепям между асинхронными тактовыми сигналами.

СОКРАЩЕНИЕ ВРЕМЕНИ НА ОТЛАДКУ

Часто даже небольшое исправление исходных RTL-кодов или ограничений требует проведения повторного синтеза и связанного с этим многочасового ожидания. Покажем, как можно уменьшить число итераций при проектировании, используя комбинацию иерархического подхода "разделяй и властвуй" и функции CoE (Continue on Error) – продолжение выполнения синтеза при возникновении ошибки, что, возможно, позволит обнаружить несколько ошибок за одну итерацию.

Поблочное проектирование – ключевая методология, направленная на ускорение синтеза и сокращение общего времени проектирования. Этот подход также позволяет сохранять и использовать уже готовые и отлаженные блоки в процессе итеративного синтеза проекта. Инструмент проектирования, поддерживающий иерархическую методологию, дает возможность создавать в какой-то степени изолированные друг от друга части проекта – так называемые точки компиляции (compile points). Некоторые инструменты также позволяют превратить части проекта, например содержащие

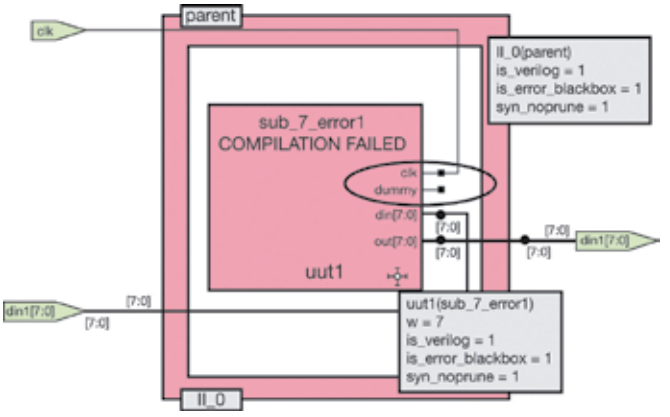

Рис.3. Графическое представление модуля с ошибками

ошибки, в "черные ящики" и экспортировать их в отдельные независимые подпроекты для дальнейшего изучения и отладки. Как только эти подпроекты будут полностью отлажены, их вновь объединяют с основным проектом либо как готовые принципиальные схемы, либо в виде RTL (используется подход "снизу вверх" или "сверху вниз", соответственно, возможен и смешанный подход).

Для интеграции и отладки больших проектов важно иметь возможность обнаруживать ошибки спецификации как можно раньше. Функция CoE позволяет получать обобщенный отчет об ошибках для каждого прохода синтеза. Она дает возможность продолжить синтез, если встречаются некритические ошибки, не связанные с синтаксисом HDL, либо ошибки соответствия (mapping) с технологической библиотекой. В итоге разработчики могут проанализировать большую часть проекта за один проход синтеза с ошибками.

Для активации функции CoE в инструментах Synplfy Pro/Premier необходимо поставить флажок Continue-on-Error с левой стороны окна проекта (Project View) или выполнить команду интерпретатора TCL

```
set _ option -continue _ on _ error 1
```

Модули с ошибками и родительские модули компонентов, имеющие ошибки согласования интерфейсов, помечаются при помощи атрибута is_error_blackbox=1 и подсвечиваются (рис.3).

Для того чтобы найти все блоки с ошибками при помощи TCL и присвоить список переменной, используется следующая команда:

```
c _ list [find -hier -inst * -filter @
is _ error _ blackbox==1]
```

А чтобы вывести список ошибочных блоков на экран, нужно выполнить:

```
get _ prop -prop inst _ of [find -hier -inst * -filter
@is _ error _ blackbox==1]
```

Модули с ошибками, которые могут быть превращены в "черные ящики" и экспортированы для независимой компиляции, обозначены на рис.3 красным цветом.

ЭКСПОРТ МОДУЛЕЙ КАК ИНСТРУМЕНТ ИЗОЛЯЦИИ ОШИБОК

Пользователь имеет возможность экспортировать модули, содержащие ошибки, в независимые проекты, которые для более детальной проверки могут быть синтезированы отдельно от основного проекта. При экспорте создается проект, в него включаются все необходимые для автономного синтеза модуля исходные HDL-файлы, подключаемые HDL- и IP-библиотеки с правильным порядком путей поиска файлов. Как говорилось выше, модули с ошибочным описанием автоматически помечаются при помощи специальных атрибутов, что позволяет с легкостью их обнаруживать и в дальнейшем экспортировать.

Для экспорта модулей нужно выбрать требуемый модуль в окне просмотра иерархии проекта или RTL, правой кнопкой мыши вызвать контекстное меню и выбрать пункт Generate Dependent File List (рис.4).

По завершении отладки блок может быть вновь встроен в основной проект либо в виде RTL-описания, которое в дальнейшем подлежит синтезу в контексте всего проекта (подход "сверху вниз"), либо как уже готовая схема ("снизу вверх") (рис.5).

Непростой задачей при использовании иерархического подхода к синтезу может стать выполнение требований по максимальным задержкам сигналов. Решить ее в значительной степени позволяет бюджетирование – создание временных бюджетов для отдельно синтезируемых блоков. Некоторые инструменты автоматически создают бюджеты времени в процессе ручного деления проекта для независимого синтеза (создание пользовательских точек компиляции). В Synplify Pro/Premier, кроме этого, имеется механизм создания автоматических точек компиляции, например, для эффективного использования многопроцессорных

серверов. В процессе бюджетирования для каждого раздела создаются так называемые интерфейсные логические модели блоков (ILM), с помощью которых оценивается влияние этого раздела на остальную часть проекта без загрузки полного описания блока либо при отсутствии этого блока. Применяя пользовательские точки компиляции, разработчик имеет возможность заменять автоматически создаваемые бюджеты на пользовательские ограничения для каждой точки.

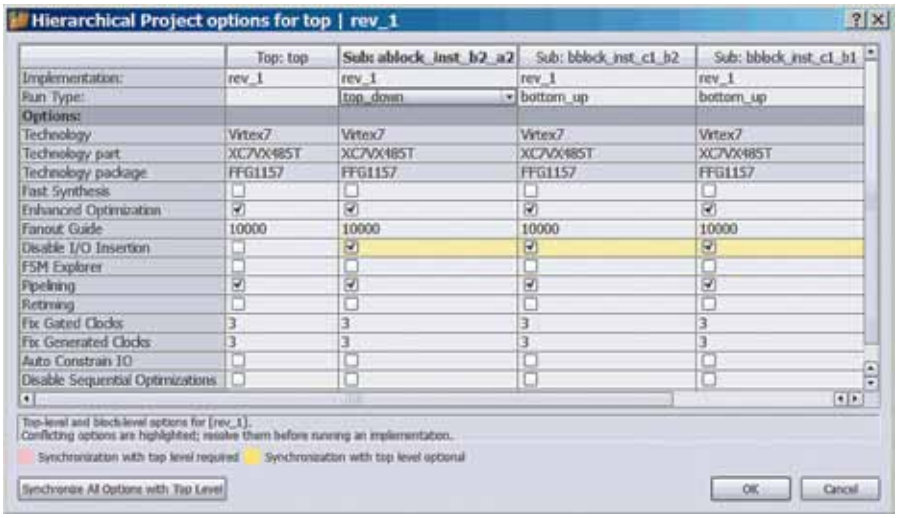

Рис.5. Включение блока в проект

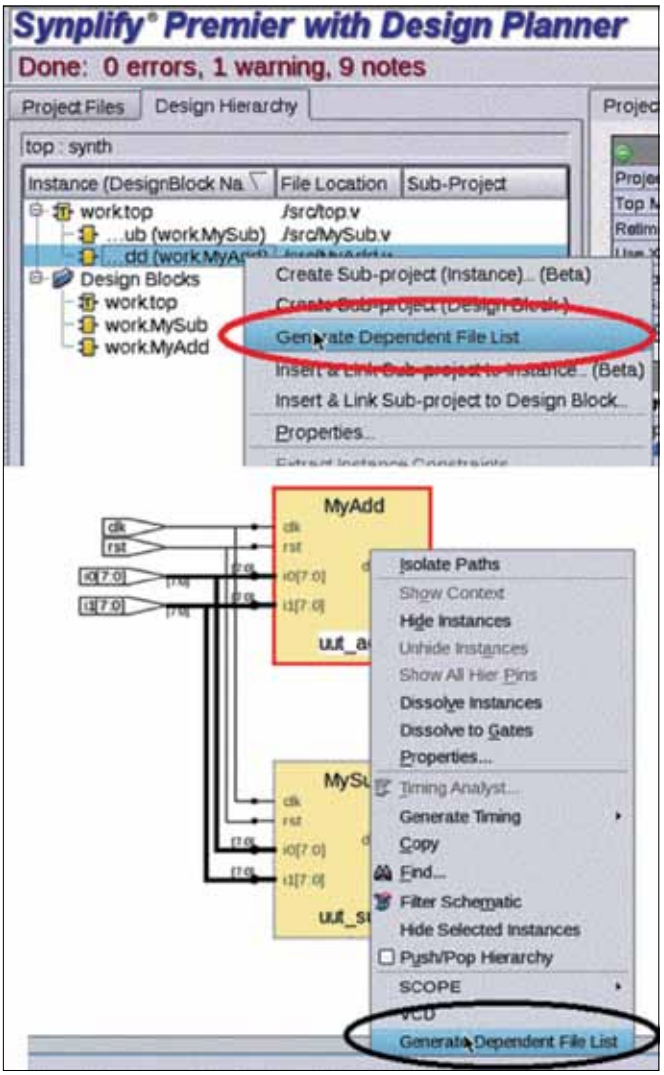

Рис.4. Экспорт модулей с ошибками

* * *

Результаты недавнего опроса пользователей, проведенного компанией Synopsys, показывают, что 59% разработчиков считают одной из наиболее трудных задач при проектировании "создание адекватной спецификации проекта". Неудачная спецификация может привести к задержке реализации проекта, а в худшем случае – к неверному функционированию. Инструменты проектирования должны иметь развитые возможности отладки, позволяющие оперативно обнаруживать и локализовывать ошибки спецификации, что позволит разработчику быстро их исправить. Неплохим дополнением к ним может стать встроенный механизм подсказок, когда инструмент сам предлагает возможные варианты исправления обнаруженных ошибок.