Продолжение. Начало в № 3`2009

Изучаем Active-HDL 7.1. Урок 9. Инструменты, повышающие эффективность работы с BDE-проектами

Александр ШАЛАГИНОВ
shalag@vt.cs.nstu.ru

Многолистовые схемы

Пакет **Active-HDL 7.1** поддерживает проектирование схем, размещенных на нескольких страницах. На самом деле это одна большая схема, которую для удобства обработки, демонстрации и восприятия нарисовали в виде нескольких фрагментов и поместили каждый на своем листе. Но заметьте, все листы по-прежнему хранятся в одном файле.

Во многих САПР для организации межстраничных связей используются специальные графические объекты, называемые соединителями страниц (английское название — offpage). В данной среде проектирования эта проблема решается гораздо проще — связыванием проводников по имени (урок 8). Другими словами, цепи, переходящие с одной страницы на другую, должны иметь одинаковое имя (рис. 1). В этом случае необходимость в соединителях страниц и вовсе отпадает.

Что касается позиционных обозначений символов, то они должны быть уникальными и не дублироваться на разных листах схемы. Впрочем, сам **BDE**-редактор следит за соблюдением этого правила.

Обычно редактор схем показывает только одну выбранную страницу схемы. Однако при желании вы можете открыть несколько страниц в отдельных окнах **BDE**-редактора или на нескольких закладках.

На рис. 1 видно, что имя активной страницы заключено в кавычки и показано в строке заголовка вслед за именем файла, где хранится многолистовая схема. В нашем примере страницы имеют имена “A” и “B”.

Что касается информации о номере текущей страницы, то она приводится на фоне общего числа страниц в статусной строке (внизу справа), например **Page 1/2** или **Page 2/2**.

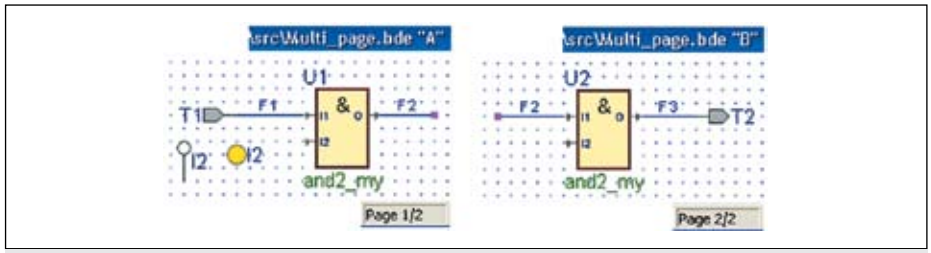


Рис. 1. Проводники F2 на разных страницах логически связаны общим именем, что равносильно их электрическому соединению

Для управления многостраничным проектом в меню **Diagram** имеется несколько команд, объединенных в разделе **Multiple pages**. Однако удобнее пользоваться не ими, а специальными кнопками, расположенными в правом нижнем углу окна схемного редактора (рис. 2а). С такой организацией навигации мы уже неоднократно встречались (уроки 4 и 5).

С помощью кнопок ◻ и ◻ можно пролистывать схему вперед и назад, а щелчком на кнопке ◻ вызвать окно со списком страниц, на которых размещена схема (рис. 2б).

В верхней части окна расположена небольшая инструментальная панель, содержащая пять пиктограмм. С их помощью вы можете добавить ◻ новую или удалить ◻ текущую страницу схемы, задать или изменить имя выделенной страницы ◻, а также добавить новую схему (в том числе и многостраничную) из другого файла ◻.

Мы уже упоминали о возможности наблюдать на экране монитора сразу несколько страниц схемы. Эта работа выполняется с помощью пиктограммы ◻ **Open in New Windows** («Открыть в новом окне»).

В процессе проектирования и отладки схем удобно пользоваться двумя вспомогательными инструментами, которые предлагают нам разработчики пакета **Active-HDL 7.1**.

Программа просмотра объектов Objects View

Программа называется **Objects View** («Просмотр объектов»). Этот инструмент показывает в удобной и компактной форме информацию обо всех объектах, используемых в схеме.

Активизировать его можно несколькими способами: из меню **View** (команда **Objects**) или щелчком по иконке ◻ на инструментальной панели **Diagram Items – VHDL**.

При запуске программы вы увидите окно, содержащее семь закладок (рис. 3), на которых размещена объектно-ориентированная информация о графических элементах схемы: параметрах настройки (**Generics**), клеммах (**Terminals**), компонентах (**Components**), сигналах (**Signals**), экземплярах (реализациях) компонентов (**Instances**), графических процессах (**Processes**) и назначениях сигнала (**Signal assignments**).

Несомненное достоинство данного инструмента — его интерактивность. Как только

Рис. 2. Кнопки и команды управления многолистовым проектом

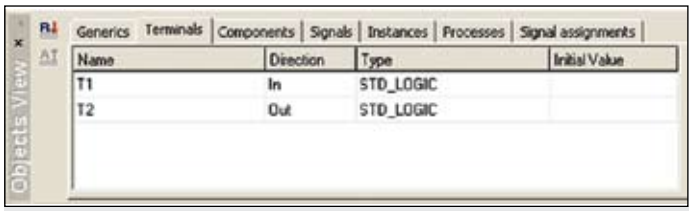


Рис. 3. Диалоговая панель вспомогательного инструмента проектирования схем Objects View («Просмотр объектов»)

вы введете очередной графический объект схемы, информация о нем тут же появится на соответствующей закладке. И наоборот, как только вы выделите объект в окне **Objects View**, он тут же высветится на схеме красным цветом.

Важно, что этот механизм работает и для многолистовых схем, загружая в редактор именно ту страницу, на которой находится искомый объект.

Слово **View** в названии программы не совсем точно отражает ее назначение. Дело не ограничивается только просмотром содержимого схемы. Вы можете сортировать объекты, изменять их свойства и даже прослеживать цепочки прохождения сигналов.

Сортировку можно выполнять в автоматическом режиме по алфавиту (в прямом и обратном порядке) или вручную методом **drag & drop**. Конечный порядок отсортированных объектов будет использован в процессе автоматической конвертации схемы в **VHDL**-код.

Но для того, чтобы изменения были переданы в схемный документ, необходимо обязательно нажать на кнопку **Apply current order to Block Diagram** («Применить текущий порядок к схеме»).

Если вы передумали и захотели восстановить прежний порядок сортировки, то потребуется другая кнопка — **Get order from Block Diagram** («Получить порядок из схемы»). Правда, положительный результат будет достигнут только в том случае, если вы еще не сохранили документ. Иначе восстановить старые настройки можно будет только командой **Undo**.

Дважды щелкнув **ЛКМ** на любом объекте или выполнив команду **Properties** из контекстного меню, вы можете «на лету» отредактировать его свойства.

Но самое интересное — это возможность просмотреть пути прохождения сигналов от входных до выходных портов и в обратном направлении.

Чтобы все получилось, вы должны предварительно инициализировать режим моделирования (команда **Initialize Simulation** из меню **Simulation**).

Отслеживать можно только объекты, расположенные на закладках **Terminals** и **Signals**, то есть клеммы и сигналы. Контекстное меню любого объекта на этих закладках содержит команду **Follow Object** («Отслеживать объект»).

Например, для входного порта T1 (рис. 1) вы увидите в контекстном меню только ссылку на сигнал-приемник . В ней указывается номер строки **VHDL**-кода, где сообщается, какой сигнал снимает данные с входного порта **T1**. Щелкните **ЛКМ** по данной ссылке, и в окне **HDL**-редактора появится соответствующий фрагмент кода: .

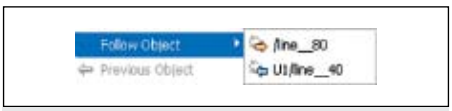


Рис. 4. Обнаружение ссылок на элементы схемы

Теперь ясно, что приемником является сигнал **F1**. Находим это имя на закладке **Signals** и повторяем операцию.

Для следующего объекта (сигнала **F1**) обнаруживаются две ссылки: одна — на элемент-последователь (**U1/line_40**), другая — на элемент-предшественник (**/line_80**) (рис. 4).

Последнее означает, что появилась возможность двигаться как по пути распространения сигнала, так и в обратном направлении, реализуя алгоритм обратной трассировки (**back trace**).

Окно запроса Query Window

Второй вспомогательный инструмент носит название **Query Window** («Окно запроса»). Основное его назначение — находить на схеме символы, иерархические блоки и цепи (включая клеммы, глобальные соединители и символы питания). Кроме того, он позволяет прослеживать связи между названными элементами схемы.

Вторая функция здесь реализована более грамотно, чем в ранее рассмотренной программе. Во-первых, теперь не нужно активизировать процесс моделирования, во-вторых, в путях прохождения сигналов участвуют все графические объекты, а не только клеммы и цепи. Это делает процесс более наглядным и понятным.

Вызвать данный инструмент можно из меню **View** (команда **Query**), щелчком по иконке, расположенной на инструментальной панели **Diagram Items – VHDL** или просто нажав горячую клавишу **Q**.

Структура окна **Query** показана на рис. 5. В верхней его части расположена инструментальная панель, с помощью которой настраиваются параметры поиска.

Нажав соответствующую пиктограмму, вы можете включить в поиск различные виды соединений (проводники, шины, клеммы, символы питания и глобальные соединители), символы и иерархические блоки, а также все их конкретные реализации.

Поиск можно выполнять с учетом регистра (при нажатой кнопке). В этом случае символы **and2** и **AND2** — это разные объекты. Если нажата кнопка, то в поиске будут участвовать только целые слова.

Чтобы отыскать конкретный элемент на схеме, надо ввести в окно поиска его имя и нажать кнопку **Find** («Найти»). На рис. 5 таким элементом является входная клемма **T1**.

Вид графического объекта и его имя высвечиваются в верхней части информационной области на голубом фоне. Затем идет сообщение о номере и имени страницы, на которой находится данный объект. На рис. 5 на желтом фоне приводится список соединений найденного объекта. В данном примере клемма **T1** имеет один контакт, и он подключен к цепи с именем **F1**.

Обратите внимание, оба имени в списке соединений подчеркнуты. Это обычные гиперссылки, позволяющие вызвать соответствующий объект. Щелкнем по ссылке **F1**, и содержимое окна запроса **Query** обновится (рис. 6а).

Видно, что проводник **F1** связывает входную клемму **T1** с символом **and2_my**, точнее с его конкретным экземпляром **U1**. Последнее в строке слово **I1** указывает имя конкретного контакта, который соединен с проводником **F1**.

Слово **U1** здесь тоже подчеркнуто, значит — это гиперссылка. Отправившись по ней, мы увидим информацию об элементе-приемнике сигнала **F1** (рис. 6б).

Помимо уже известной информации, выясняется, что он находится в библиотеке **Lesson_BDE**. Символ имеет три контакта с именами (**Pin Name**) **I1**, **O**, **I2**. Выход **O** подсоединен к проводнику **F2**. Щелкнув по ссылке **F2**, определим, куда идет этот провод. Ситуацию проясняет третий слайд (рис. 6в). Он как раз и демонстрирует соединение по имени, связывая выход **O** компонента **U1** с входом **I1** компонента **U2**. Первый расположен на странице "A", второй — на странице "B".

На рис. 5 есть еще пара «загадочных» кнопок: **Regular Expression** и. При нажатой кнопке включается режим поиска объектов с использованием регулярных выражений. Они бывают полезны, когда требуется создать какой-то особенный (необычный) шаблон поиска. Например, вы хотите отыскать на схеме

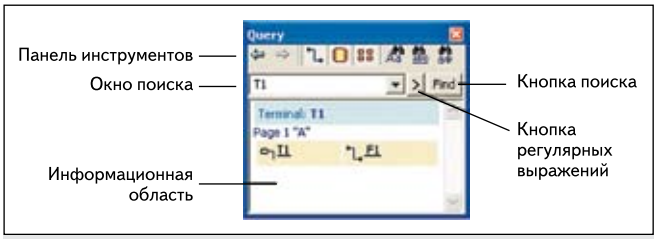


Рис. 5. Окно вспомогательного инструмента Query Window для поиска элементов схемы и отслеживания соединений между ними

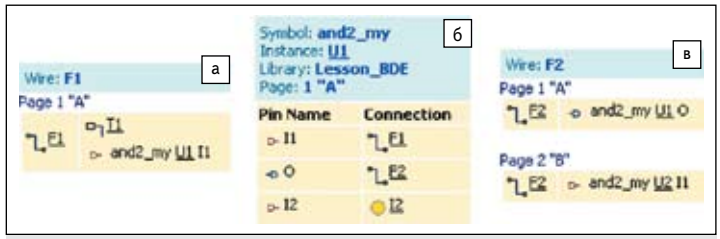


Рис. 6. В окно запроса теперь выводится информация о новом выделенном объекте

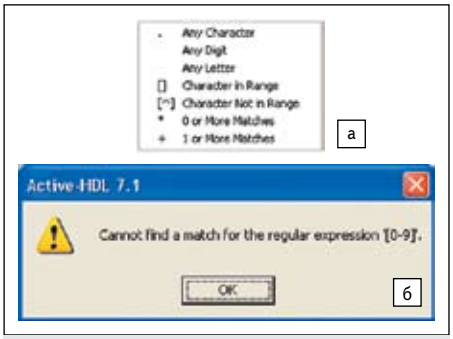


Рис. 7. а) Список заготовок для создания регулярных выражений; б) сообщение о неудачном завершении поиска

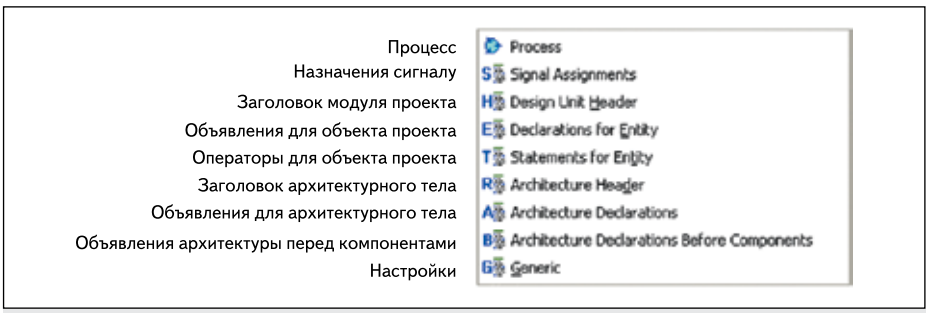


Рис. 8. Команды меню Diagram/VHDL для вызова девяти типов HDL-блоков

какую-нибудь цепь питания **VCC** или **GND**. Введя в окно поиска текст vcclgnd, вы легко решите поставленную задачу. Вертикальная черта, служащая разделителем между словами vcc и gnd, является одним из таких регулярных выражений.

Щелчком по кнопке раскрываем список заготовок (рис. 7а), помогающий формировать часто используемые регулярные выражения. Если поиск заканчивается неудачей, на экране монитора появляется соответствующее сообщение (рис. 7б).

Мы не станем вдаваться в подробности работы этого механизма поиска, так как для неискушенного пользователя он не представляет особого интереса.

Работа с текстовыми HDL-блоками

В процессе создания схемы иногда возникает необходимость добавить к автоматически генерируемому **HDL**-коду текстовую информацию, например, библиотеку, пакет или какой-нибудь **HDL**-оператор. Оказывается, это можно сделать прямо на схеме с помощью специальных текстовых блоков, содержащих **HDL**-код.

Самое интересное заключается в том, что редактор следит за тем, чтобы добавляемый текст был вставлен в нужный раздел структуры **HDL**-кода.

Для языка **VHDL** существует девять типов таких блоков (рис. 8). Правда, один из них, называемый графическим процессом, занимает особое положение. С ним мы уже немного знакомы на примере мультиплексора **mux2** (урок 7, рис. 8).

Названные блоки можно вызвать из меню **Diagram/VHDL** или с помощью пиктограмм инструментальной панели **Diagram Items – VHDL**. Пиктограмма **Add Process** нам уже знакома, а потому мы оставим ее без комментариев.

Чтобы получить доступ к любому из восьми оставшихся блоков, нужно раскрыть выпадающий список, щелкнув по небольшому треугольнику, расположенному справа от изображения ранее использованного блока, и выбрать желаемый тип (рис. 9).

Обычно **HDL**-блоки используются как вспомогательное средство, но в принципе и целую схему можно представить в подобном стиле.

На такой «схеме» (рис. 10) вы не обнаружите ни одного символа и ни одного проводника, ничего, кроме **HDL**-блоков. И все же это будет структурное описание проекта.

Приступая к проектированию схемы на текстовых **HDL**-блоках, следует помнить, что шаблоны объекта проекта и его архитектурного тела будут созданы автоматически. Более того, к ним автоматически будет добавлен стандартный заголовок **Design unit header** со ссылкой на библиотеку **IEEE** и стандартный пакет **std_logic_1164**. Все остальное описание схемы придется создавать вручную.

Прежде всего, с помощью блока **Entity Declarations** необходимо определить входные и выходные порты мультиплексора **D0**, **A**, **D1** и **Y**, а в блоке **Architecture declarations** объявить внутренние сигналы схемы — **F1**, **F2** и **F3**.

Далее, вызвав блок **Architecture declarations before components** (рис. 10), следует указать, какие компоненты будут использованы в схеме. В данном примере мы собираемся применить только один библиотечный элемент — инвертор **inv_my**.

Текст, который вы видите в этом блоке, совсем не обязательно вводить с клавиатуры. Проще поступить так. Активизировать закладку **Library Manager**, выбрать на ней рабочую библиотеку **lesson_bde** и отыскать там нужный компонент **inv_my**. Щелкнув по нему **ПКМ**, выполнить команду **Copy Declaration**. Теперь остается только вставить скопированную декларацию в **HDL**-блок.

Все, с объявлениями покончено. Настала пора описать структуру схемы мультиплексора. Компоненты схемы задаются с помощью текстового **HDL**-блока **Signal Assignments**. Для библиотечного элемента inv_my содержимое блока тоже можно скопировать из менеджера библиотек так, как мы это только что делали.

Правда, понадобится другая команда контекстного меню, а именно **Copy VHDL Instantiation**. Карту порта вставленного экземпляра объекта inv_my тоже придется отредактировать, так как его входной контакт подключен в схеме к цепи **A**, а выходной — к цепи **F1** (см. урок 1 или 7). Заодно изменим и заданное по умолчанию название метки **Label1** на стандартное имя **U1**.

Элементы **U2** и **U3**, выполняющие функцию 2И, представим в потоковой форме с помощью еще одного блока — **Signal Assignments_2**. Элемент **U4**, выполняющий функцию 2ИЛИ, опишем в виде графического процесса **or2_my_1**.

Для того чтобы вручную сформировать список чувствительности этого процесса, придется сбросить флажок

Рис. 9. Выпадающий список

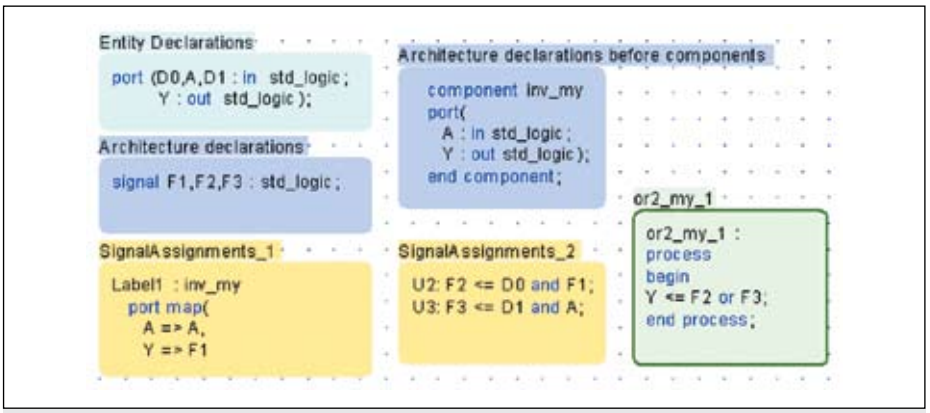


Рис. 10. Изображение схемы мультиплексора mux2 в виде текстовых HDL-блоков

на закладке **Sensitivity List** диалоговой панели **Process Properties**. Иначе вам придется подводить к блоку все проводники, с которых процесс читает значения сигналов.

Созданную «схему» следует обязательно откомпилировать, а затем и промоделировать, так как велика вероятность, что мы что-нибудь пропустили. Подозреваем, что проделанная работа не вызвала у вас особого энтузиазма, скорее вы испытали определенное разочарование, так как вместо привычного схемотехнику занятия пришлось «погружаться» в проблемы программирования.

Но это только пример, который не является образцом для подражания, а подчеркивает лишь широкие возможности пакета **Active-HDL 7.1**.

Проверка правильности проекта

В пакете **Active-HDL 7.1** есть инструмент, который позволяет выяснить, имеются ли в созданной схеме ошибки. Он называется **Design Rule Check** — контроль правильности проекта (сокращенно **DRC**).

По умолчанию данная программа запускается в автоматическом режиме, как только вы подадите команду откомпилировать свою схему или создать для нее **VHDL**-код. В зависимости от степени серьезности программа **DRC** делит все нарушения на две категории: ошибки (**errors**) и предупреждения (**warnings**).

Ошибки — это серьезные нарушения, и их надо устранять в обязательном порядке, иначе проектируемое устройство будет неправильно работать, и, вероятнее всего, его просто не удастся смоделировать.

Обнаружив ошибку, система **Active-HDL 7.1** просто откажется генерировать **HDL**-код. Впрочем, вы можете проявить настойчивость и «заставить» ее выполнить данную работу. Для этого нужно открыть диалоговую панель **Code Generation Setting** и на закладке **Generation** сбросить флажок **Check diagram before code generation** («Выполнять контроль схемы перед генерацией кода»).

Предупреждения, то есть незначительные нарушения в схеме, не приводят к фатальному исходу. При их наличии в схеме система продолжит работу и конвертирует схемное описание в **VHDL**-код, хотя и предупредит об их присутствии. Другими словами, предупреждения целиком и полностью лежат на вашей совести.

Заметим, что **DRC**-проверке подвергается только открытый в настоящее время схемный файл. В нем может содержаться и многолистовая схема. А вот для иерархической схемы вложенные описания автоматически обрабатываться программой **DRC** не будут.

Создадим схемный файл и назовем его **DRC_error**. Нарисуем в нем несколько фрагментов абстрактной схемы (рис. 11) с типичными ошибками, которые совершает обычно начинающий пользователь.

Первый фрагмент (рис. 11а) содержит сразу две ошибки. Проводник **S(4)**, являющийся отводом от шины, имеет имя, лежащее вне диапазона индексов шины. Другое ответвление вообще не имеет имени. И то, и другое — грубые ошибки.

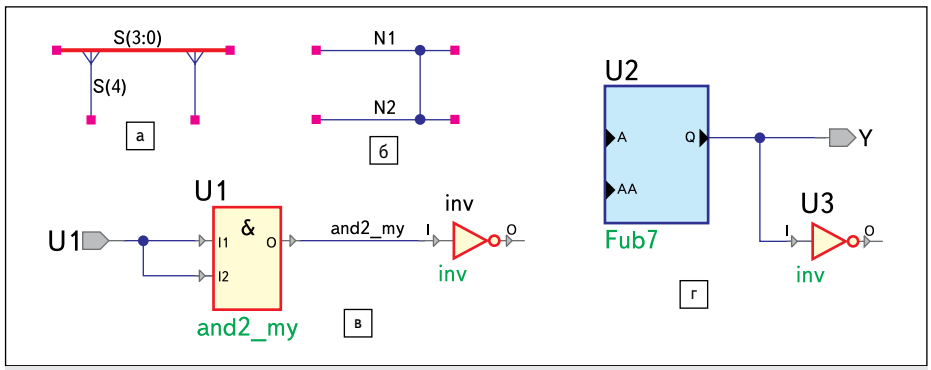


Рис. 11. Двухстраничная схема, содержащая множество ошибок

Понятно, что имена проводников, подключенных к шине, должны быть членами этой шины. В данном примере они должны называться **S(0)**, **S(1)**, **S(2)** и **S(3)**.

Второй пример (рис. 11б) демонстрирует закороченные цепи (**shorted nets**). Цепи считаются закороченными только в том случае, если они электрически соединены и в то же время имеют различные пользовательские имена. На неименованные цепи данное правило не распространяется.

Третий фрагмент (рис. 11в) показывает нарушение правил присваивания имен графическим объектам схемы. Нельзя, в частности, задавать одинаковые имена портам и позиционным обозначениям (в данном примере имя **U1**), сигналам и символам (имя **and2_my**), символам и позиционным обозначениям (имя **inv**).

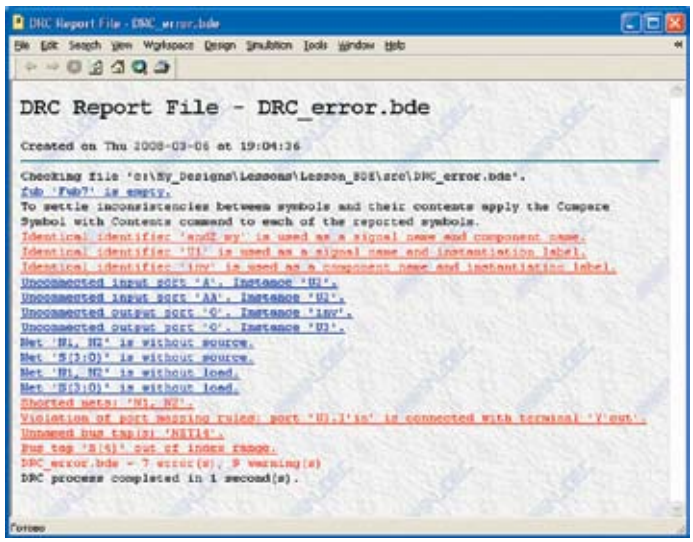
На рис. 11г показана еще одна часто встречающаяся, но совсем не очевидная ошибка. Выход **Q** иерархического блока **Fub7** связан безымянной цепью с выходным портом **Y** (здесь ошибки нет) и с входным контактом символа **U3**. Казалось бы, и здесь нет никаких нарушений.

Но система считает иначе. В этой ситуации она регистрирует соединение входа символа **U3** с выходным портом **Y**, что противоречит синтаксису языка **VHDL** для карты порта.

Для устранения обнаруженного системой последнего дефекта есть несколько способов, но проще всего назначить неименованной цепи какое-нибудь пользовательское имя, отличное от имени выходного порта.

Наиболее очевидные ошибки отслеживает сам **BDE**-редактор, причем эту работу он выполняет в интерактивном режиме. Например, попытку ввести повторяющиеся имена для позиционных обозначений символов или контактов иерархического блока редактор просто откажется выполнять.

То же самое касается ввода имен, не соответствующих требованиям, предъявляемым к идентификаторам языка **VHDL**. Вас ждет неудача, если вы попробуете задать какому-либо объекту схемы имя **entity**, **out** или **in**. Ясно, что зарезервированные слова для имен не годятся.


DRC Report File - DRC_error.bde

Created on Thu 2008-03-06 at 19:04:36

Checking file 'd:\My_Designs\Lessons\Lesson_80E\src\DRC_error.bde'.
Fub 'Fub7' is empty.
To settle inconsistencies between symbols and their contents apply the Compare Symbol with Contents command to each of the reported symbols.
Identical identifier 'and2_my' is used as a signal name and component name.
Identical identifier 'U1' is used as a signal name and instantiation label.
Identical identifier 'inv' is used as a component name and instantiation label.
Unconnected input port 'A', Instance 'U2'.
Unconnected input port 'AA', Instance 'U2'.
Unconnected output port 'O', Instance 'inv'.
Unconnected output port 'O', Instance 'U3'.
Net 'N1, N2' is without source.
Net 'S(3:0)' is without source.
Net 'N1, N2' is without load.
Net 'S(3:0)' is without load.
Shorted nets: 'N1, N2'.
Violation of port mapping rules: port 'U2.I'in' is connected with terminal 'Y'out'.
Unnamed bus tap(s) 'NET14'.
Bus tap 'S(4)' out of index range.
DRC_error.bde - 7 error(s), 3 warning(s)
DRC process completed in 1 second(s).

Рис. 12. Отчет о результатах проверки схемы, изображенной на рис. 11

Служебные символы тоже нельзя использовать в качестве имен. Например, имена **out-1** или **A>B** будут забракованы схемным редактором.

Сведения об ошибках и предупреждениях программа **Design Rule Check** записывает в отдельный **htm**-файл, который автоматически открывается в отдельном окне **DRC Report File** (рис. 12).

Уточним, вы увидите отчет о результатах **DRC**-контроля только в том случае, если в схеме найдены ошибки. В остальных ситуациях результаты **DRC**-проверки можно вывести на экран, выполнив двойной щелчок мышью на строке **Double click on this line to view the log**, которая появится в окне **Console** после окончания процедуры.

Отчет можно переадресовать на консоль, если на панели **Check Diagram Settings** установить флажок **Output to console**.

Заметим, что уровень серьезности каждого нарушения в схеме тоже можно изменять с помощью диалоговой панели **Check Diagram Settings**.

В настройках разрешается выбрать любое из трех значений: **Ignore**, **Warning** и **Error** (рис. 13). В принципе любому нарушению в схеме можно при желании задать один из названных уровней серьезности: игнорировать, предупреждение, ошибка. Однако принятые по умолчанию установки представляются вполне разумными, и вряд ли вам потребуется их изменять.

Рис. 13. Выбор настроек

Продолжение следует