

Продолжение. Начало в № 3`2009

# Изучаем Active-HDL 7.1.

## Урок 11. Как проектировать иерархические блоки

Александр ШАЛАГИНОВ  
shalag@vt.cs.nstu.ru

### Размещение иерархических блоков

Чтобы разместить иерархический блок на схеме (назовем ее `test_fub.bde`), достаточно выполнить команду **Diagram/Fub**, щелкнуть по соответствующей иконке **Fub (F)** или нажать на горячую клавишу **F**. Курсор мыши приобретет форму графического изображения компонента, показывая таким образом, что редактор готов рисовать иерархический блок.

Нажмите **ЛКМ** в том месте, где вы хотите поместить один из углов блока и, не отпуская кнопку, передвиньте мышшь в противоположный по диагонали угол. Отпустите кнопку, когда размеры блока будут соответствовать

вашим желаниям. Особой тщательности здесь не требуется: габариты блока можно изменить позднее.

Подведите к иерархическому блоку проводник. В точке, где проводник встретится с контуром блока, появится входной контакт. Если вы начнете рисовать проводник из блока, то будет сгенерирован выходной контакт (рис. 1).

Неудобство описанного способа задания выводов заключается в том, что они получают системные имена подключенных к ним проводников. Поэтому их придется редактировать.

Если же сначала нарисовать проводники и назначить им имена (рис. 2а), то при последующем размещении блока все получится наилучшим образом (рис. 2б). С тем же успехом можно использовать внешние выводы схемы — терминалы (**terminals**) (рис. 2в).

Удалите какой-нибудь проводник, подключенный к блоку, и вы обнаружите, что исчезнет и его одноименный контакт.

Закончив формирование интерфейса, следует отредактировать стандартное имя **Fub1**, задаваемое по умолчанию. Дважды щелкнем

на нем мышкой и введем новое название, например `mux2_schema_fub`.

Обязательно сохраните схему с нарисованным иерархическим блоком, только тогда он появится в рабочей библиотеке (рис. 3). Если удалить или просто отсоединить (**Detach**) схему от проекта, то из библиотеки исчезнет и ИБ.

Следовательно, ИБ может существовать только с момента размещения его на схеме. Более того, на схеме ИБ может присутствовать только в одном экземпляре, вот почему вы не обнаружите подобные объекты в «ящике символов».

Еще раз подчеркнем: создавая блок, нам даже не пришлось переключаться в режим редактирования ИБ. Между тем такая возможность имеется.

### Редактирование иерархических блоков

Чтобы перейти в режим редактирования, надо щелкнуть на блоке **ПКМ** и в контекстном меню выбрать команду **Edit**. Познакомимся с этим способом. Нарисуем пустой ИБ (рис. 4а) и перейдем в режим редактирования (рис. 4б). Если не удастся перевести редактор в данный режим, значит, вы забыли сохранить схему.

В режиме редактирования фон рабочей области становится серым, на экране монитора автоматически появляется окно **Add New Pins** для добавления новых контактов (рис. 5), а редактируемый блок обрамляется контуром с кривой штриховкой.

Нам потребуются три входных и один выходной контакты. Они передаются в проектируемый блок методом **Drag and Drop**. По умолчанию контакты получают стандартные имена `Pin1... Pin4` (рис. 4в), поэтому их придется отредактировать с помощью команды **Properties** из контекстного меню (рис. 4г).

Для окончания редактирования следует щелкнуть **ЛКМ** за пределами блока и утвердительно ответить на предложение сохра-

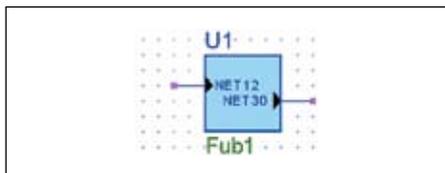


Рис. 1. Иерархический блок

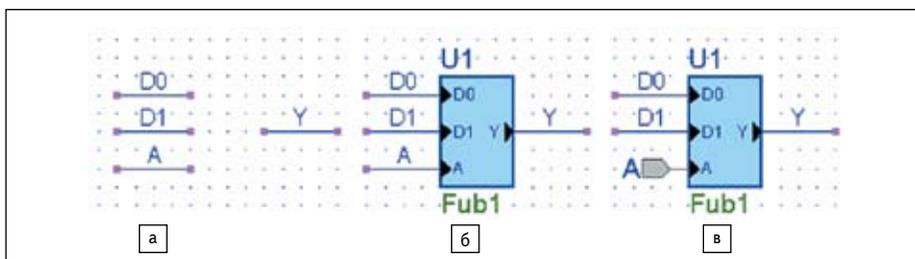


Рис. 2. При размещении ИБ на схеме контур блока должен касаться концов проводников или терминалов

Library	Unit Name	Secondary Unit Name	Source Type	Target Language	Symbol	Simulation Data
lesson_11	mux2_schema_fub	N/A	N/A	Unspecified	Fub	No

Специальная метка Fub указывает на то, что сохраненный в рабочей библиотеке объект является иерархическим блоком

Рис. 3. В рабочей библиотеке проекта появился созданный иерархический блок `mux2_schema_fub`

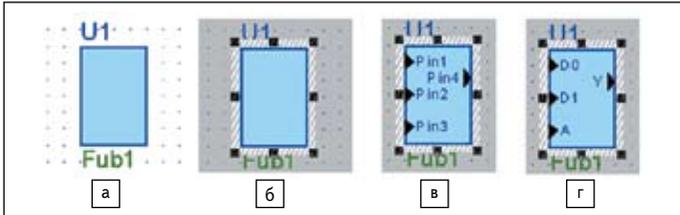


Рис. 4. Этапы проектирования иерархического блока

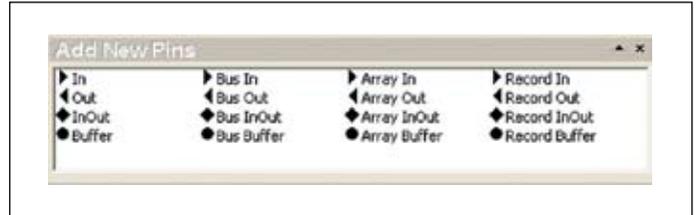


Рис. 5. Окно Add New Pins, содержащее все типы контактов иерархического блока

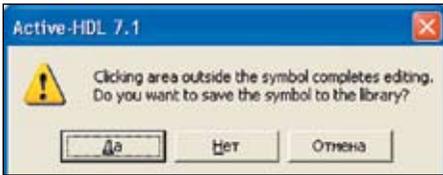


Рис. 6. Заканчивая редактирование, сохраним ИБ в рабочей библиотеке, нажав на кнопку «Да»

нить созданный объект в рабочей библиотеке (рис. 6).

На схеме полученный иерархический блок (рис. 7а) должен быть подключен к проводникам. Проще всего такую работу выполнить, используя команду **Add Stubs** из контекстного меню. Редактор подсоединит небольшие отрезки проводников ко всем свободным контактам ИБ и присвоит им те же самые имена, которыми названы подключенные контакты (рис. 7б). Если теперь удалить подходящий к ИБ проводник, то вы не потеряете связанный с ним контакт.

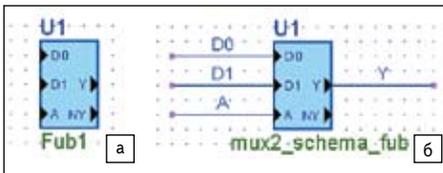


Рис. 7. Подключаем проводники к свободным контактам ИБ командой Add Stubs

### Проектирование внутреннего содержимого иерархического блока

Итак, графический образ блока есть, теперь займемся его «начинкой». Двойным щелчком по блоку вызовем диалоговую панель **Create New Implementation** (рис. 8) и определим способ внутреннего описания блока. Другими словами, нам предстоит решить вопрос о том, как будет реализован (**implementation**) проектируемый блок. Остановимся на схемном описании блока (**Block Diagram**).

Как только вы нажмете на кнопку **OK**, автоматически будет создан и открыт схемный файл **mux2\_schema\_fub.bde**. Остается лишь нарисовать схему мультиплексора, на что вам понадобится 3–4 минуты (урок 1).

Закончив рисование схемы, щелкните пару раз в любом свободном месте, и редак-

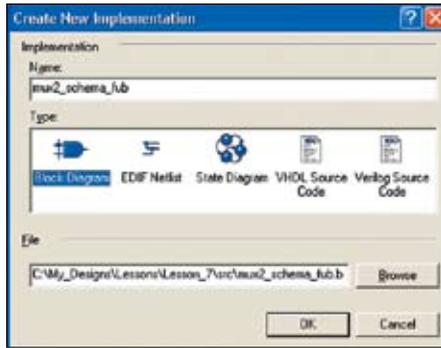


Рис. 8. В качестве внутреннего описания (реализации) блока выбираем схему

тор повысит уровень иерархии, снова покажет блок как черный ящик. Тот же результат будет достигнут, если вы щелкнете по иконке **Hierarchy Pop**, находящейся на инструментальной панели **BDE Edit**.

Промоделируйте схему, содержащую фактически только один иерархический блок (рис. 7б), чтобы убедиться, что все работает правильно.

А теперь внесем в проект небольшие изменения. В процессе проектирования, основанном на методе «проб и ошибок», такой работы не избежать.

Допустим, нам потребовался, кроме прямого, еще и инверсный выход мультиплексора. Придется добавить к иерархическому блоку **mux2\_schema\_fub** дополнительный выходной контакт (рис. 9а). Понятно, что изменения следует внести и в содержимое блока, в данном случае в его схемное описание.

Но как это лучше сделать? Предлагаем вариант с привлечением панели **Compare Interface** (рис. 10). Она активизируется соответствующей командой **Compare Symbol with Contents...** Щелкните по блоку ПКМ и вы увидите ее в контекстном меню.

Выделенная серым цветом строка (рис. 10) показывает обнаруженные различия и предлагает внести изменения в схемный файл. Об этом говорит переключатель, установленный в положение **Update file mux2\_schema\_fub.bde**. В нижнем левом углу можно также получить информацию о числе несоответствий между внешним и внутренним описаниями блока. В нашем примере найдено одно различие: **Differences in 1 port(s) found**.

Нажав на кнопку **OK**, вы обнаружите, что в схемное описание блока добавился выходной порт **NY**. Осталось лишь модифицировать схему под свои нужды, например, так, как это сделано на рис. 9б.

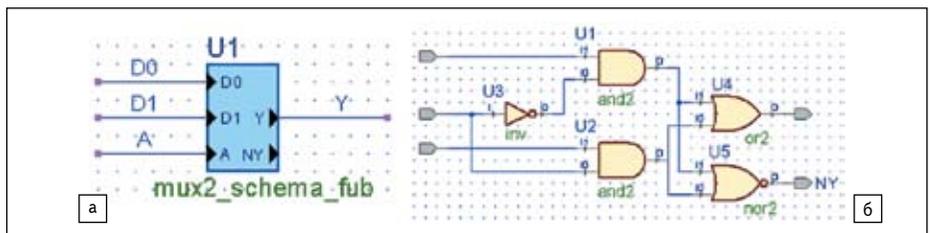


Рис. 9. Вносим в схему мультиплексора необходимые изменения (добавляем элемент U5)

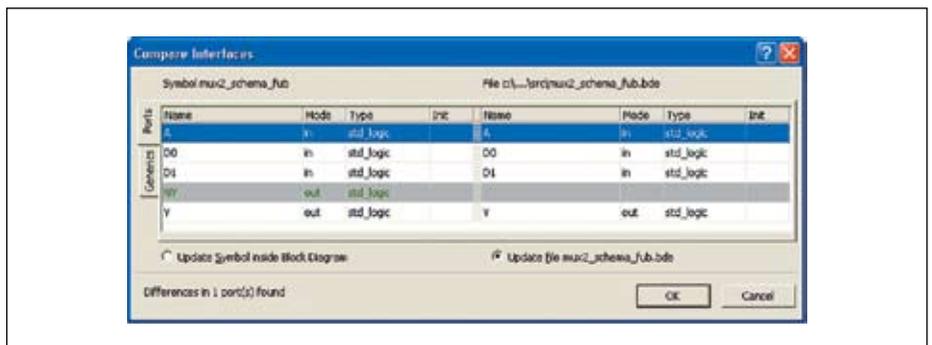


Рис. 10. Сравниваем порты иерархического блока и его схемного описания

Рассмотренный вариант построения иерархического блока был выполнен по технологии **top-down** (сверху вниз). Но оказывается, можно действовать и в противоположном направлении, то есть сначала создать внутреннее описание блока, а затем его внешний вид (**bottom-up**).

Подтвердим сказанное примером. Напишем VHDL-модель того же мультиплексора (рис. 11) и сохраним ее в файле **mux2\_HDL\_fub.vhd**.

Затем нарисуем на схеме **test\_fub.bde** еще один иерархический блок (без контактов), дважды щелкнем на нем мышью и зададим тип внутреннего описания — **VHDL Source Code**. Система предложит создать VHDL-файл с именем **Fub1.vhd**.

Но мы не станем следовать этому совету, а нажмем кнопку **Browse** на панели **Create New Implementation** и сделаем ссылку на только что созданный файл **mux2\_HDL\_fub.vhd**.

Обнаружив, что такой файл существует в данном проекте, редактор запросит подтверждение на его использование — **Use existing file** (рис. 12). Это как раз и входит в наши планы.

Получив подтверждение, редактор автоматически расставит контакты на иерархическом блоке и заменит его стандартное имя (рис. 13).

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux2_HDL_fub is
  port (
    D0 : in STD_LOGIC;
    D1 : in STD_LOGIC;
    A  : in STD_LOGIC;
    Y  : out STD_LOGIC;
    NY : out STD_LOGIC
  );
end mux2_HDL_fub;

architecture mux2_HDL_fub of mux2_HDL_fub is
begin
  Y <= (D0 and not A) or (D1 and A);
  NY <= (D0 and not A) nor (D1 and A);
end mux2_HDL_fub;
```

Рис. 11. Потокное VHDL-описание работы мультиплексора с парафазным выходом

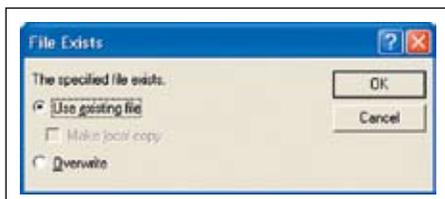


Рис. 12. Подтверждаем использование существующего файла

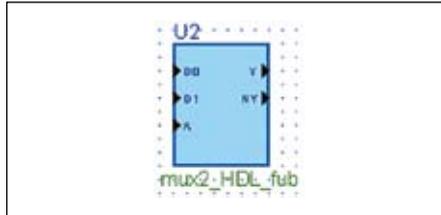


Рис. 13. Замена имени иерархического блока

Раньше нам приходилось это делать вручную. Выполните операцию **Push**, чтобы убедиться, что с данным блоком связан именно тот файл, который мы и собирались подключить в качестве его содержимого (рис. 11).

Ранее мы говорили о том, что на схеме любой блок может существовать только в одном экземпляре, поэтому в «ящике с символами» иерархические блоки не показываются в буфер обмена какой-нибудь из двух созданных нами иерархических блоков и вставляем в ту же самую схему.

Вы обнаружите, что редактор повторил только внешний вид блока, заменив уникальное имя стандартным **Fub1**. При попытке понизить уровень иерархии выяснится, что внутреннего описания у «копии» и вовсе нет. Значит, это не копия, а новый иерархический блок с таким же интерфейсом, как у оригинала.

Однако если вы попытаете прикрепить к новому блоку тот же файл, что и у оригинала, то получите реплику, говорящую о том, что такой блок уже используется в данной схеме (рис. 14).



Рис. 14. Попытка поместить на схеме копию иерархического блока заканчивается неудачей

Более того, неудачей закончится и попытка вставить копию в другую схему того же проекта.

А причина всех неудач объясняется просто. В рабочей библиотеке (одной на проект) уже зарегистрирован ИБ с именем оригина-

ла, так что копию с тем же именем система просто откажется добавлять в библиотеку.

Другими словами, в рамках одного проекта не может быть двух ИБ с одинаковыми именами или со ссылкой на одну и ту же реализацию.

## Внутреннее описание иерархического блока на языке EDIF

Существует еще одна реализация ИБ — в виде списка цепей (**Netlist**) на языке **EDIF**. Скорее всего, вы уже обращали на него внимание. Например, при выборе способа реализации блока (рис. 8) вторая слева позиция предлагает как раз вариант **EDIF Netlist**.

Аббревиатура **EDIF** расшифровывается как **Electronic Design Interchange Format** — «формат обмена проектами при разработке электронных схем». Список цепей в этом стандарте может быть получен из описаний проекта на языках **VHDL** или **Verilog** с помощью стандартных программ или непосредственно из схемы.

Любой серьезный пакет просто обязан поддерживать этот международный стандарт, и **Active-HDL 7.1** не является здесь исключением. В качестве примера назовем использование языка **EDIF** для описания компонентов, входящих во многие системные библиотеки этой среды проектирования.

Описание на языке **EDIF** имеет несколько признаков. Мы познакомимся с ними на примере элемента **sop4**, входящего в библиотеку **spartan3** (рис. 15).

Из графического изображения элемента видно (рис. 15а), что **sop4** выполняет функцию **2-2И-2ИЛИ**. В системной библиотеке **spartan3** информация о данном элементе представлена следующей строкой (рис. 15б).

Обратите внимание, в столбце **Target Language** указан целевой язык описания **EDIF**. Кроме того, в столбце **Unit Name** перед именем модуля **sop4** стоит буква **D**, что тоже является признаком **EDIF**-кодирования.

И, наконец, если вы загрузите в **Active-HDL 7.1** библиотеку **spartan3** как проект, то в окне просмотра **Design Browser** обнаружите значок с той же буквой **D** . Этим значком система помечает все библиотечные компоненты с **EDIF**-описанием.

Однако попытка создать иерархический блок со списком цепей на языке **EDIF** окажется неудачной, если вы предварительно



Рис. 15. Общие сведения о компоненте **sop4** из системной библиотеки **spartan3**

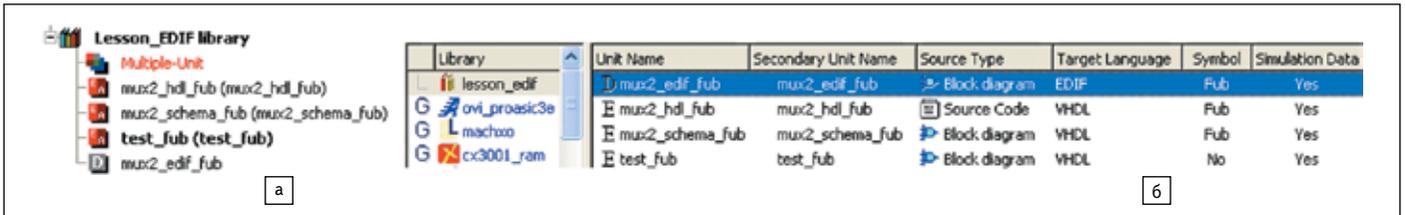


Рис. 19. Для всех модулей проекта имеются данные для моделирования Simulation Data = Yes: а) фрагмент окна Design Browser; б) фрагмент окна менеджера библиотек Library Manager

не подготовили такой список. Зная, что список **Netlist** представляет собой текстовую форму кодирования схемы, мы можем попробовать получить его, конвертируя схему, подобно тому, как система автоматически генерировала **VHDL**-список цепей из графического описания проекта (рис. 9а).

Это продуктивная идея, поэтому попробуем ее осуществить. Запустим **BDE**-редактор и создадим новый схемный файл **mux2\_EDIF\_fub.bde**. В процессе его создания укажем рабочий язык **EDIF** (рис. 16). Очень важный момент!



Рис. 16. Для новой схемы выбираем другой целевой язык — EDIF (вместо установленного по умолчанию языка VHDL)

Копируем в него старую схему из файла **mux2\_schema\_fub.bde** и компилируем его. В окне просмотра проекта **Design Browser** должен появиться новый файл **mux2\_EDIF\_fub.edn** (рис. 17) со списком цепей в формате **EDIF**, а в рабочей библиотеке — его откомпилированный модуль **mux2\_edif\_fub**.



Рис. 17. Просмотр проекта Design Browser

Откройте названный файл и убедитесь, что имеете дело с языком **EDIF**. В самом начале файла вы увидите текст:

```
1 | edif mux2_EDIF_fub
2 | (edifVersion 2 0 0)
```

Первое слово в этом тексте — язык описания, последнее — его версия.

Надо сказать, что описание схемы на языке **EDIF** хотя и имеет текстовый вид, однако

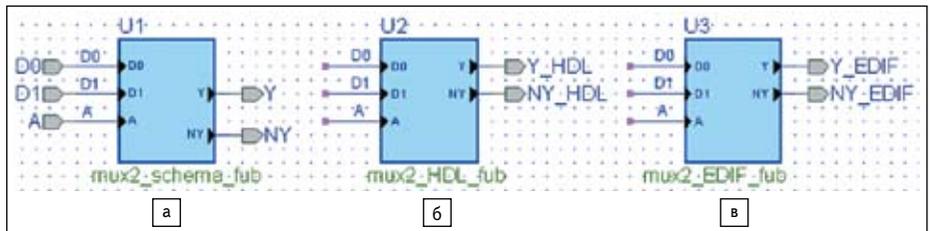


Рис. 18. Иерархические блоки с разными формами внутреннего описания проекта: в виде схемы, HDL-кода и EDIF списка цепей

для ручного кодирования не предназначено. Списки цепей на языке **EDIF** генерируются и используются программами САПР в автоматическом режиме, а потому погружаться в детали описания вряд ли имеет смысл.

Впрочем, обнаружить в этом описании элементы схемы и связи между ними не представляет особого труда.

Теперь, когда имеется файл со списком цепей, остается лишь подключить его к иерархическому блоку **U3** (рис. 18) и откомпилировать все модули.

Активизируйте еще раз закладку **Library Manager** и убедитесь, что в рабочей библиотеке имеются все необходимые модули (три иерархических блока и схема испытаний **test\_fub**). Причем каждый модуль должен иметь данные для моделирования **Simulation Data = Yes** (рис. 19). Промоделируйте свой проект, чтобы рассеялись последние сомнения.

Впрочем, кое-что так и осталось неясным. Почему, например, при двойном щелчке **ЛКМ** по иерархическому блоку **U3** (рис. 18) открывается не список цепей **EDIF**, а схема, из которой он был сгенерирован? Ведь мы присоединили к блоку текстовый, а не графический файл.

И последнее. Иногда возникает необходимость поменять реализацию блока. Можно ли это сделать? Конечно, и очень просто. Для этого надо щелкнуть по блоку **ПКМ**, выполнить команду **Properties...**, а затем на открывшейся панели **Fub Properties** ввести в поле **Fub name** другое имя, являющееся ссылкой на новую реализацию.

Можно много экспериментировать с иерархическими блоками и получать любопытные, а порой и просто неожиданные результаты, однако вряд ли «овчинка стоит выделки». Проще конвертировать созданный блок в символ, и тогда он станет доступным в любых проектах.

### Преобразование иерархических блоков в символы

Операция эта очень простая и выполняется одной командой **Convert Fub to Symbol**, вызываемой из контекстного меню. На рис. 20 показан результат преобразования иерархического блока **mux2\_schema\_fub** в символ.

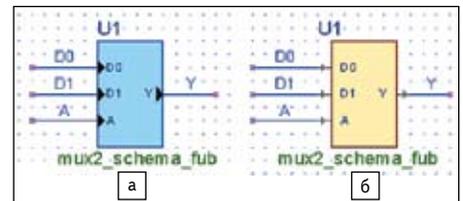


Рис. 20. Преобразование иерархического блока в символ

Выполняя данную операцию, следует помнить, что, получив символ, вы навсегда потеряете конвертируемый иерархический блок. Даже всеисильная команда **Undo** не поможет вам вернуться к старому ИБ.

*Продолжение следует*