Окончание. Начало в № 5`2006

Система проектирования Altium Designer 6

Владимир Пранович

Pranovich@bsu.by

Подходы и примеры реализаций типовых схемных и топологических решений

Создание классов цепей

Классы цепей можно создать:

- непосредственно для всего проекта;
- на схеме с помощью директив (TR0111 Schematic Editor and Object Reference);
- с помощью средства **Board Definition & Rules** PCBDOC редактора (TR0104 Altium Designer Panels Reference).

Автоматическое создание классов всего проекта предусмотрено для всех цепей, принадлежащих:

- одному **BUS** (цепей с одноименными названиями и переменной цифровой частью);
- одному компоненту;
- группе компонентов с одинаковым параметром **ClassName**;
- всем **NET** с одинаковым параметром **ClassName** (данный параметр, относящийся к **BUS**, считается параметром для всех **NET**, принадлежащих этой **BUS**);
- одному листу схемы проекта.

Большой выбор автоматического создания классов существенно облегчает способ их определения, и для малых проектов достаточно ограничиться этим набором (рис. 26).

Однако не следует злоупотреблять установкой везде «галок», так как это приведет к огромному множеству классов, с которыми вы с трудом разберетесь. В наших примерах мы вообще не будем прибегать к автоматическому созданию правил.

Задание классов цепей непосредственно в схеме можно сделать с помощью директив **Place\Directive\ Net Classes**. При этом определяется имя класса и в параметрах задаются свойства цепей, принадлежащих данному классу. Как правило, подобный способ задания цепей можно рекомендовать, когда проект ведется разными группами разработчиков для схемного редактора и непосредственно работающих с трассированием в PCBDOC. При этом именно разработчик схемы определяет наиболее важные группы цепей и их параметры. В этом случае и имя класса, и его параметры можно сделать видимыми на схеме. Однако они обычно громоздки и не информативны при отображении на твердой копии схемы. Типовая ошибка пользователей, переходящих на данный пакет, состоит в том, что они сразу начинают прописывать классы прямо на схеме. В примерах, приведенных далее, мы избежим и этого, так как задавать классы и их параметры таким способом достаточно утомительное и (в начале изучения пакета и проектирования первых проектов) бесполезное дело.

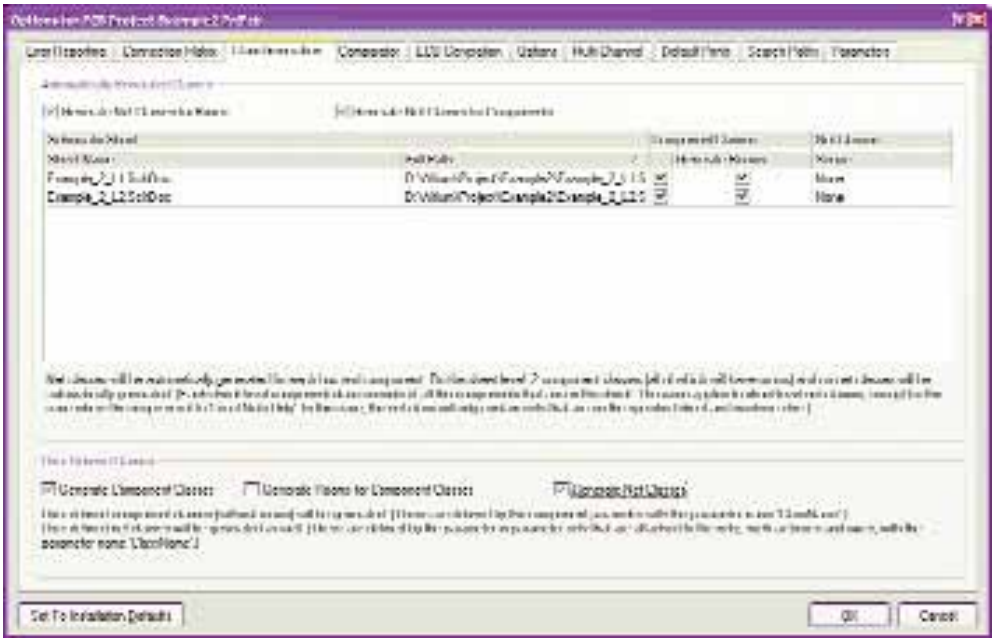

Рис. 26. Окно автоматического задания классов цепей

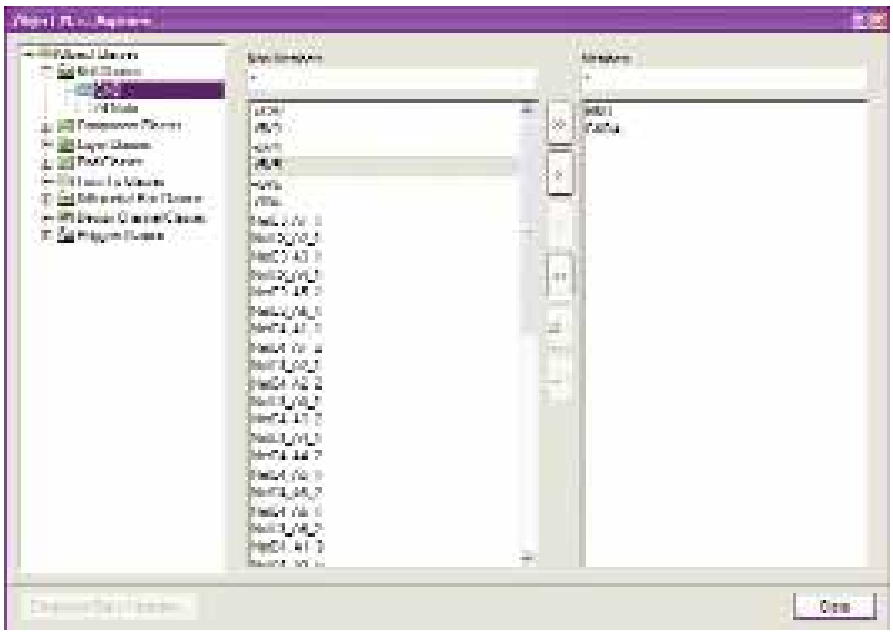

Рис. 27. Окно определения классов цепей

Итак. Определим классы цепей непосредственно в PCBDOC редакторе командой **Design/Classes**. Для нашей схемы введем 2 класса цепей, а именно **GND** и **Power**. Для этого откроем ветку **NetClasses** дерева **ObjectClasses** и щелчком правой кнопки мышки выберем из меню **AddClasses**. При этом у нас сразу добавится новый класс с именем **NewClass**. Аналогично предыдущему, находясь на **NewClass**, выберем из меню **RenameClasses** и введем имя нашего нового класса **GND**. При выделенном классе **GND** отметим в окне **NonMember NET** с именами **GNDA** и **GND** и перенесем их с помощью кнопок в окно **Member**. Таким образом, мы определили класс **GND** и его наполнение и окно **Object class explorer** (рис. 27).

Аналогично добавим класс **Power** и его наполнение. Пользователи легко в дальнейшем определятся и с другими типами классов цепей.

Создание правил для цепей и классов

В примере (см. ТвЭП № 5–6 `2006) при проектировании платы мы задавали пределы для ширины дорожек и зазоров общие, для всех случаев. Теперь попробуем задать разные правила для различных классов цепей, а также другие типовые и наиболее часто употребляемые правила при проектировании печатных плат. В данной статье мы не будем останавливаться на специфических и редко используемых правилах. Не будем также подробно комментировать, почему выбран тот или иной параметр для правила и его значение.

Вызов панели задания правил производится командой **Design Rules**.

Правила Electrical

Рассмотрим одно из самых употребительных правил **Electrical/Clearance**, определяющее минимальные зазоры между электрическими объектами (дорожками, переходными отверстиями и другими объектами), задающими проводящий рисунок платы. Сейчас у нас прописано единственное правило, установленное нами ранее и действующее на все цепи. Для рассмотрения типовых подходов при создании новых правил добавим (пусть и не совсем подходящих нашему примеру) 2 новых:

1. Зазор между цепями класса **Power** и другими объектами на слое **TOP** установим равным не менее 0,4 мм.
2. Зазор между цепями класса **GND**, принадлежащими **PAD** прямоугольного типа и объектами типа **VIA** или **PAD,** установим равным не менее 0,3 мм.

Аналогично добавлению нового класса выделим новое правило и нажатием правой кнопки мыши выберем **NewRule**. При этом добавится новое правило **Clearance_1**. Выделим его и переименуем в **Power_0,4mm**. Данное правило имеет тип BINARY и определяет правила между двумя типами объектов. В колонке **Where the first…** отмечаем **Net Class** и в выпадающем параметре выбираем наш класс **Power**. В колонке **Where the second…** отмечаем **Lays** и в выпадающем списке параметров выбираем слой **Top Layer**. Устанавливаем требуемый зазор, равный 0,4 мм. Окно правил представлено на рис. 28.

Таким образом, достаточно быстро задать все простые правила. Теперь перейдем к более сложному правилу для второго случая.

Добавим еще одно правило с именем **GND_1_0,3mm**. Однако нам не хватит типовых наборов (первых пяти) в колонках **Where the …**, поэтому воспользуемся **Advanced (Query)**. Для колонки **Where the first…** дополнительно нажмем кнопку **Query Helper**. В появившемся окне **PCB Function** выбираем **Object Type checks** и его свойство **IsNet**. После чего надпись **IsNet** перемещается в верхнюю часть окна. На панели операндов нажимаем знак «=» и сами дописываем имя цепи — **GND**. Первая часть выражения написана. Теперь на панели операндов нажимаем **And**, а в окне **PCB Function** выбираем **Object Type checks** и его свойство **IsPad**. В верхней части получаем выражение **IsNet=GND And IsPad**. Внизу нажимаем кнопку **Check Syntax**. Если появляется надпись **Expression is o'key**, мы можем принять правило. Так строятся очень сложные правила, и их следует всегда проверять на правильность применения. Для колонки **Where the second…** с целью разнообразия нажмем кнопку **Query Builder**. В левом выпадающем меню выберем условие **Object Kind Is** в правом **PAD** и в правом окош-

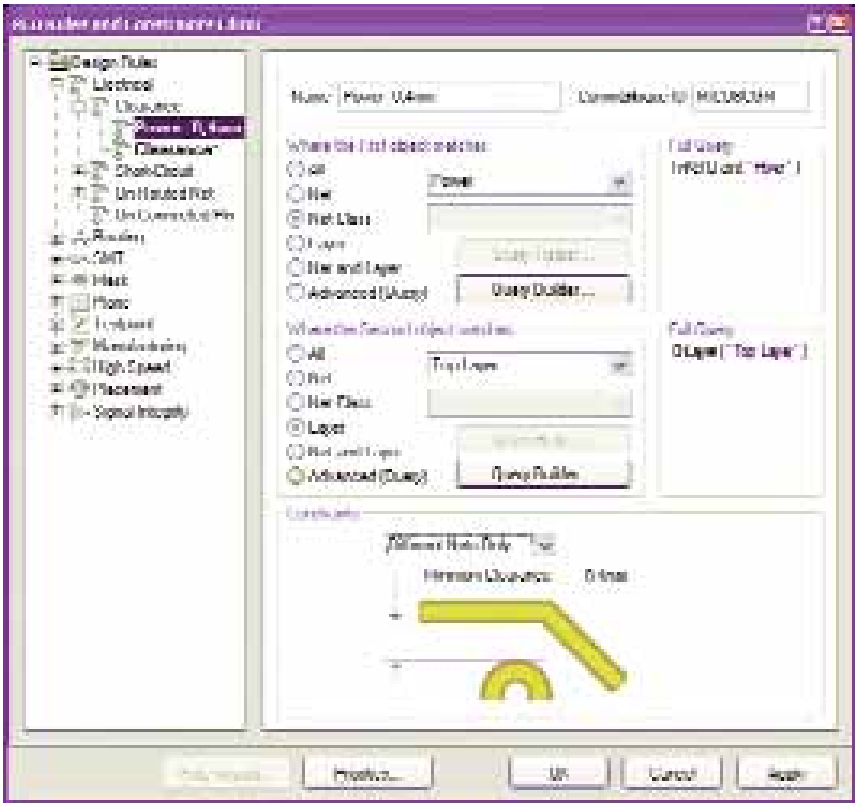

Рис. 28. Окно задания правил Electrical

Рис. 29. Окно задания правил

ке получим надпись **IsPad**. Выберем в первой колонке надпись **Add another condition**, затем **Object Kind Is** и надпись **IsVia. C**тавим оператор **OR**. В итоге формируется выражение **IsPad OR IsVia**. Здесь строить выражение проще, но меньше вариантов. Вид обоих окон представлен далее. Наконец зададим, как и для предыдущего класса, требуемый зазор и будем считать, что правила **Clearance** определены для нашего проекта (рис. 29).

Правила **Short Circuit** не требуют пояснений. Отметим, что они могут использоваться для одной физической цепи, которая на схеме может быть обозначена несколькими **NET**, а замкнута только на **PCB**. Также не требуют особых пояснений правила **UnRoutedNet** и **UnConnectedPin**.

Правила Routing

Основное правило **Width**. Правила задания свойств такие же, как и выше, за исключением того, что это правило имеет тип UNARY. Здесь пользователь может указать ширину в зависимости от слоя и т. п. Построение правил аналогичное. Единственно отличие — это правило имеет не одно, а три значения: минимальное, максимальное и установленное по умолчанию, что эффективно используется при интерактивной разводке.

Routing Topology, Priority, Layers, Corners определяют типовой набор правил, характерный и для других аналогичных пакетов, и определяют тип стратегии разводки, приоритет последовательности разводки, направления и стоимости разводки в слоях, вид округления «углов» дорожек при трассировке. Ряд правил **Fanout Control** определяет порядок и форму подвода дорожек к микросхемам для поверхностного монтажа.

Правило **Routing Vias** аналогично **Width** и так же активно используется при интерактивной разводке.

Отдельно в этом ряду стоит правило **DiffPairsRouting** для разводки дифференциальных пар. Однако оно не должно вызывать больших затруднений.

Специальная группа правил **SMT** формирует порядок подвода дорожек к SMD-элементам и связана с технологическими требованиями монтажа таких элементов: как правило, при автоматической пайке для обеспечения требуемого термобарьера; исключения «наплыва» припоя при «остром» угле подключения дорожки (близком положении угла поворота дорожки) к SMD-площадке или близком положении угла поворота дорожки.

Правила Mask

Правило **Solder Mask Expansion** определяет зазор между защитной маской и размерами **PAD**. По умолчанию зазор установлен в 0,1 мм. Пользователь вправе его увеличить и уменьшить, более того, сделать отрицательным. Последний способ можно использовать, когда, например, **PAD** увеличенного размера используется для крепления транзисторов с целью повышения теплоотвода, при этом теплоотвод обеспечивается и с учетом закрытия части **PAD** маской. Правило **Paste Mask Expansion** определяет зазор при изготовлении трафарета нанесения пасты.

Правила Plane

Правила задают способ и параметры (включая термобарьеры) подключения слоев **Plane** и полигонов к переходным отверстиям и PAD. Рассмотрим только правило **PolygonConnect**. Правило задает способ подключения PAD к полигону:

- **Relief Connect** — с термобарьером, при этом определяется число и ширина **Relief,** а также угол подвода к PAD;
- **DirectConnect** — без термобарьера;
- **NoConnect** — запрет подключения.

Обычно эти правила различны для разных полигонов и прописываются аналогично правилу **Clearance.** В нашем случае ограничимся созданием одного правила **PolygonConnect PadMultiLayer**, разрешающего подключение к полигону всех PAD для штыревых элементов, как показано на рис. 30.

Правила Testpoint, Manufacturing, High Speed, Signal Integrity

Из большой группы этих правил рассмотрим лишь несколько.

Правило **HoleSize**. Определим для нашего проекта два правила. Одно для всех цепей с минимальным (0,5 мм) и максимальным (1 мм) размером Hole, и второе — для силовых цепей, как показано на рис. 31, с увеличенными значениями Hole (для обеспечения требований максимального тока для силовых линий).

Правило **ViaUndoSMD**. Установим галочку **Allow vias undo smd Pads**. Предположим, что наша плата будет только ручной сборки, и главным требованием является плотность сборки и минимизация числа используемых слоев.

Правило **ComponentClearance**. В данном правиле оставим стандартные зазоры и установим **CheckMode=FullCheck**. Связано это с тем, что при создании посадочных мест мы не делали ни **ComponentBody,** ни контур **Courtyard** для допуска посадочных мест. В будущем рекомендуем сразу формировать данные изображения, тем более что в команде **Tools/IPC Footprint Wizard** появилась фун-

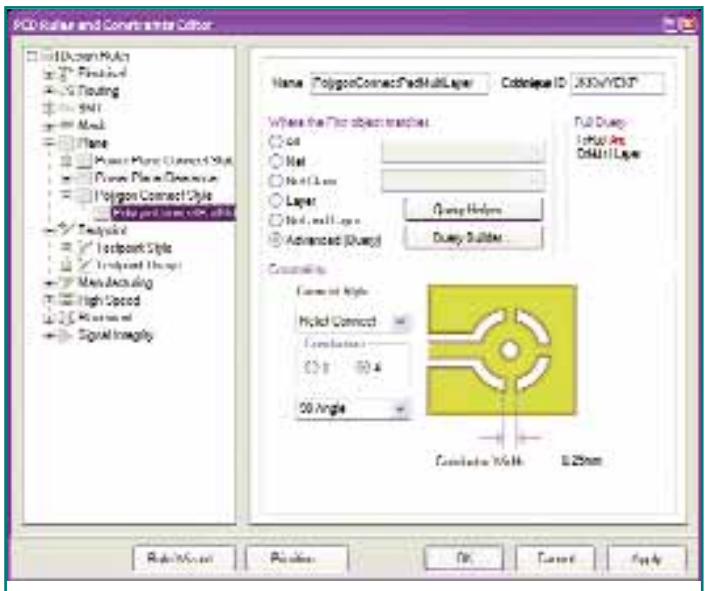

Рис .30. Окно задания правил Plane

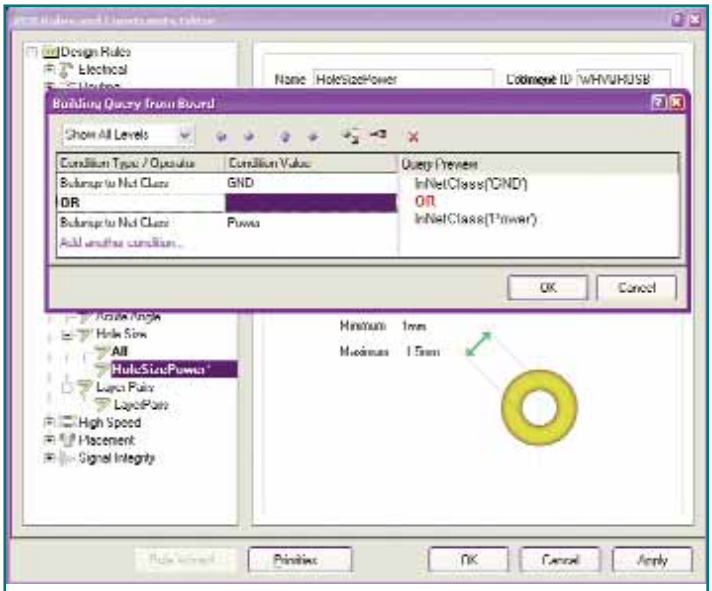

Рис. 31. Окно задания правил HoleSize

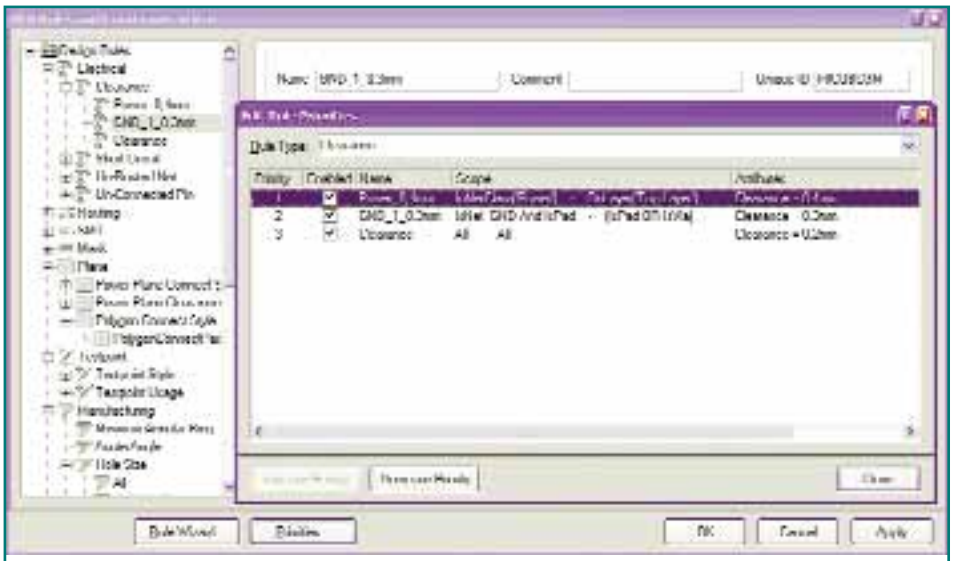

Рис. 32. Задание приоритета правил

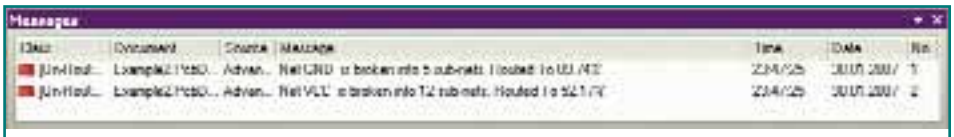

Рис. 33. Сообщения об ошибках

кция автоматического формирования данных изображений для множества типов корпусов. В этом случае можно использовать и другие значения параметра **CheckMode.** При этом проверка выполнения этого параметра будет доступна не только при запуске программы проверки, но и в интерактивном размещении, и произойдет значительно быстрее.

Параметр **Hight**. Его следует вводить и определять для специально созданных или имеющихся **ROOM.**

Задание приоритета правил и проверка их выполнения

На этом будем считать завершенным создание правил для нашего примера и перейдем к заданию приоритетов правил. Задать приоритет правил можно непосредственно в окне **PCB Rule and…** нажатием кнопки **Priorites** для определенного типа правил. Далее представлена установка таких приоритетов для параметра **Clearance** в нашем примере. Аналогично проверяем и устанавливаем порядок приоритетов для всех правил одного типа (рис. 32).

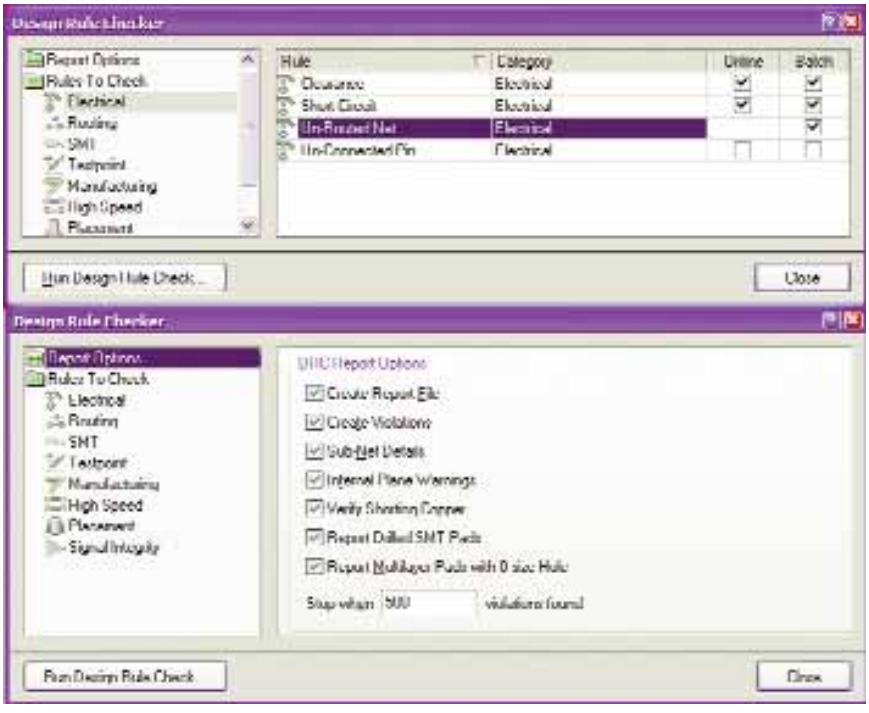

Рис. 34. Проверка выполнения требований правил

Многие правила одинаковы для разных проектов. И более того, у пользователей образуется целый свод любимых правил. Нет необходимости для каждого проекта так долго их вводить. Щелчок правой кнопкой мышки в левой части окна **PCB Rule and…** вызывает меню, позволяющее сделать экспорт или импорт выбранных правил в отдельный файл с расширением *.rul. Можно в этом файле хранить наиболее часто используемые правила и затем импортировать их в новые проекты.

Так как мы ввели не только много новых правил, но и изменили значения параметров существующих, следует проверить проект на соблюдение требований новых правил. Делаем это с помощью команды **Tolls/Design Rule Check**, установив в ней проверку нужных для нас параметров. Параметры, отмеченные в колонке **OnLine**, будут всегда проверяться при интерактивной разводке, а отмеченные в колонке **Batch** — только при запуске указанной программой. Здесь же, в окне **Report Option,** можно указать опции формирования Report-файла (рис. 34).

В результате проверки у нас есть сообщения об ошибках. Откроем панель **System/ Messages** (рис. 33).

Ни одно из новых правил не нарушено. Две ошибки свидетельствуют о незавершенности нашего проекта, когда мы делали иерархический проект. И теперь необходимо завершить разводку **NET** (а именно **GND** и **VCC**), присутствующих в нескольких иерархических блоках.

Создание полигонов (TR0112 PCB Editor and Object Reference)

Завершим разводку нашего проекта. Одновременно введем в наш пример несколько полигонов и применим разные требования к ним.

1. Первый связан с **NET=GNDA.** Главное требование — обеспечить экранизацию различных участков.
2. Второй связан с **NET=GND.** Главное требование — обеспечить максимум заливки и максимально широкие области при подводке к **PAD** с большими импульсными и постоянными значениями тока.
3. Третий связан с **NET=VCC.** Главное требование — обеспечить только максимально широкие области при подводке к **PAD** элементов, потребляющих большую мощность.

Для первого случая включим настройки, как показано на рис. 35.

При прорисовке сложных контуров полигона не всегда сразу удается правильно расположить его углы. Для доступа к изменению сложной формы полигона выделим его, затем следует нажать правую кнопку мышки и в контекстном меню выбрать **Polygon Action/Move vertex**. После этого в углах полигона (как на рис. 35) появятся их метки, которые можно переместить в требуемое положение.

Мы указали область заливки сплошной. Однако все **PAD** цепи **GNDA** оказались присоединенными к полигону. Для аналоговой

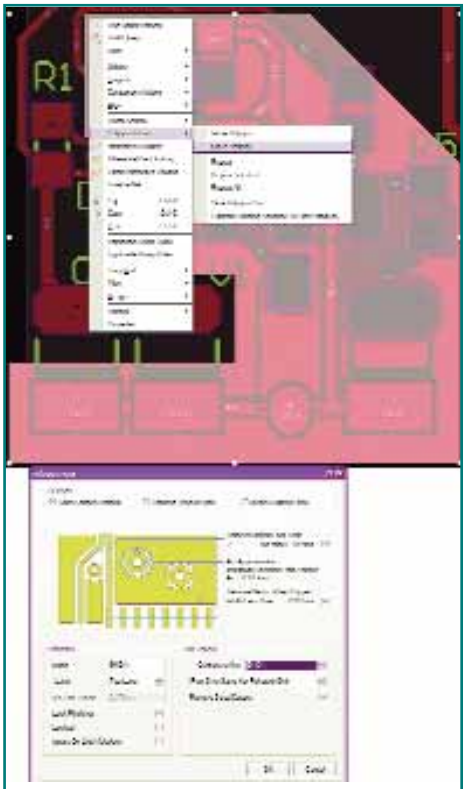

Рис. 35. Параметры полигона

«земли» желательно избежать такого соединения, а полигон подключить к данной цепи только в одной точке. Более того, данный полигон следует соединить затем и с полигоном **GND** в этой же точке. Для реализации этого придется вернуться к правилам и запретить подключение **PAD** к полигону **GNDA** одним из рассмотренных выше способов. Для подключения полигона **GNDA** к **NET** и соединения его с другим полигоном **GND** создадим новый библиотечный элемент, например, из двух **PIN**, указав его тип **Tie Net** и соединив на схеме с его помощью цепи **GNDA** и **GND,** как показано на рис. 36. При создании посадочного места для компонента **TIE** его **PAD** делаем с перекрытием или иной формы, обеспечивающей физическое замыкание между контактами.

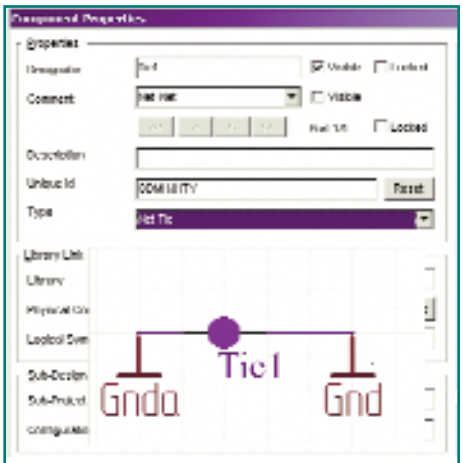

Рис. 36. Свойство компонента и подключение с его помощью разных цепей для физического объединения на плате

После внесения всех изменений наш полигон будет выглядеть так. Все соединения с **NET GNDA** произведены с помощью дорожек. Сам полигон занимает все свободное место в соответствии с установленными зазорами и подключен только в одной точке (**VIA** возле компонента **TIE1**). Объединение **NET GNDA** и **GND** произведено типологически, через **PAD** компонента **TIE1**. В этом случае любое пересечение или несоблюдение зазоров при разводке **NET GNDA** и **GND** (кроме места компонента **TIE1**) вызовет сообщение о нарушении правил трассировки.

Создание полигона для **NET GND** наиболее просто. Аналогично предыдущему делаем прямоугольный полигон на всю область нашей платы. Полигон **GND** на рис. 38 выделен более ярким цветом. Так как он делался после полигона **GNDA**, область заливки обтекает **GND** этого полигона.

Однако теперь нет места для задания третьего полигона **VCC**. Далее будет приведено решение, и не только для этого случая. Как правило, полигоны бывают большие и требуют много времени на их постоянную перезаливку. Для того чтобы полигон не мешал текущей разводке или внесению изменения, мы можем временно скрыть один или все полигоны. Для этого воспользуемся командой **Tools/Polygon Pour/Shelve…**. Вид полигонов станет неотображаемым, и мы можем приступить к формированию полигона **VCC**. Так как нам нужно только увеличить области подвода шин питания, лучше сделать это не одним полигоном, а группой идентичных. После проведения всех манипуляций мы получим удовлетворяющую нас заливку полигона **VCC**. Теперь нам важна последовательность заливки полигонов: сначала — **GNDA**, как наиболее важный и ответственный, затем группу полигонов **VCC**, для обеспечения его первичной заливки, и, наконец, **GND** — для заливки оставшейся свободной поверхности. Можно, конечно, скрыть все, и затем по очереди в необходимой последовательности открывать и снова заливать. Это просто для элементарных проектов, где число полигонов ограничено. Для проектов с большим числом разнообразных типов полигонов лучше воспользоваться **Tools/Polygon Pour/Polygon Manager** (рис. 39), где можно указать как порядок заливки полигонов, так и оперативно внести изменения в некоторые параметры. Там же показан итоговый результат проекта со всеми полигонами.

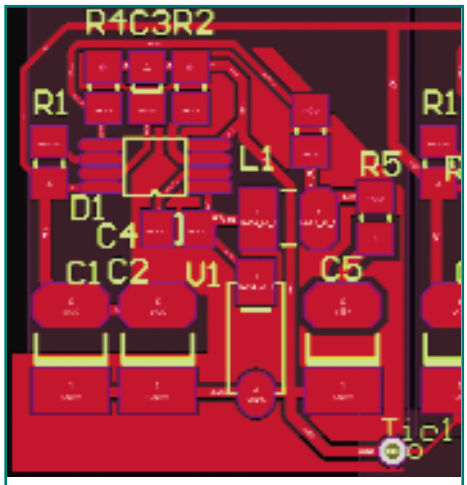

Рис. 37. Вид полигона с подключением в одной точке

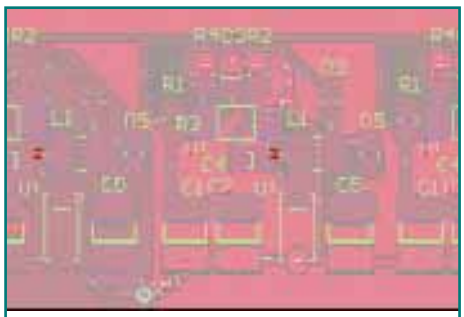

Рис. 38. Типовой полигон GND

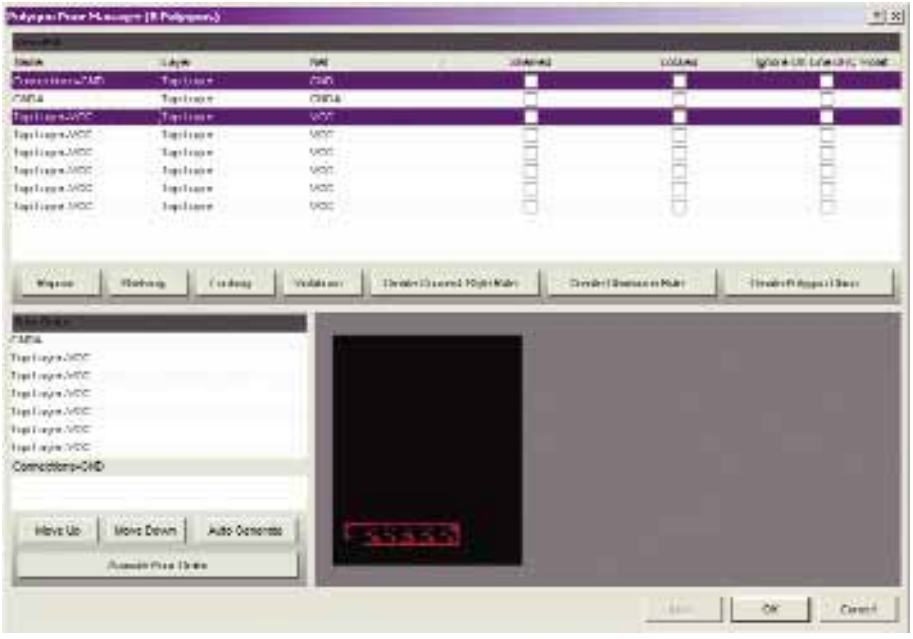

Рис. 39. Tools/Polygon Pour/Polygon Manager и итоговый вид полигонов

Применение SWAP в PCBDOC (AP0138 Pin and Part Swapping with Dynamic Net Assignment)

В связи с появлением одногейтовых компонентов логических элементов и широким распространением многослойных плат применение данной функций резко сократилось. По крайней мере, в более чем 2-летней практике автора эта функция ни разу не была востребована. Однако большинство пользователей, переходящих со старых версий других САПР, пользовались такой функцией и по традиции считают ее необходимой. В последней версии **Altium Designer 6** такая возможность появилась, и мы вкратце остановимся на ней.

Добавим в нашу схему блок, показанный на рис. 40. В блоке присутствуют компоненты с рядом эквивалентных **PIN** и **PART**.

Для задания эквивалентности **PIN** и **PART** используется команда **Tools/Configure Pin Swapping.** При этом вызывается таблица компонентов проекта, где можно выбрать компонент и затем в соответствующей таблице задать эквивалентность **PIN** и **PART** (рис. 41).

После передачи изменений в PCBDOC документ и расположения элементов с помощью команды **Tools/ Pin/Part Swapping/Configure** разрешаем указанным компонентам возможность взаимного замещения эквивалентных **PIN** и **PART**. Затем одной из команд данной группы применяем команду на применение оптимального или ручного замещения. В результате оптимизации произведена замена 9 из 11 возможных замен, а также выводится и другая справочная информация (рис. 42).

Теперь введем изменения, сделанные в PCB, в схему с помощью команды **Design/Update Shematic….**

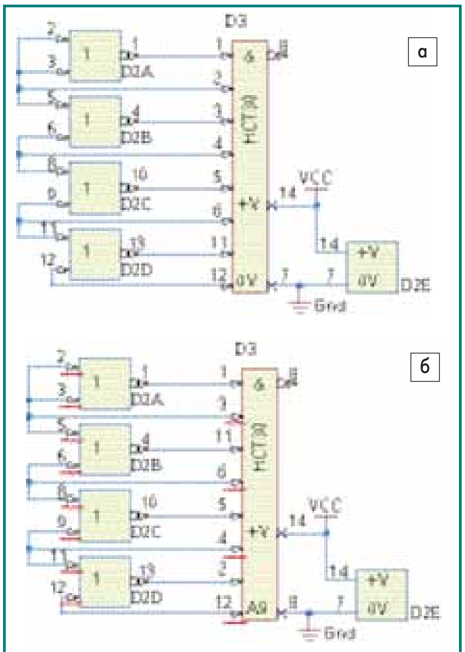

Рис. 40. Схема для демонстрации SWAP функций: а) вид до оптимизации; б) вид после оптимизации

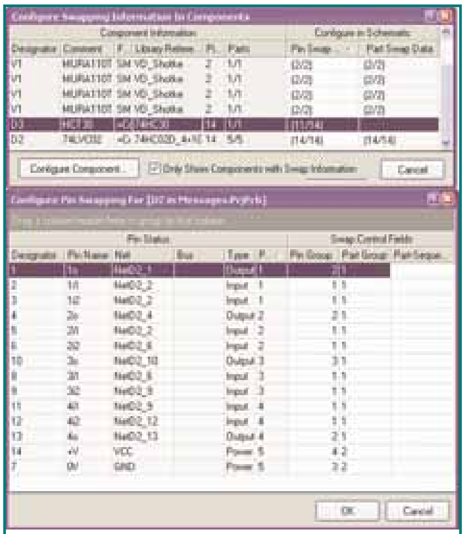

Рис. 41. Задание эквивалентности PIN и PART компонента

Базы данных (AP0134 Linking Existing Components to Your Company Database)

В заключение затронем еще одно качество САПР, которое пока редко применяется пользователями, однако оно дает прекрасную возможность для совместной работы над одним проектом, снижения ошибок при проектировании и быстром получении и модификации сопроводительной информации — это подключение баз данных. База может быть подключена:

- к проекту для автоматического переноса парамеров компонентов из базы в схему;
- инсталлирована как библиотека, для выбора самих компонентов прямо из базы.

Как правило, большинство компонентов при проектировании схемы ставится в нее без учета номиналов, вариантов посадочных мест и других параметров, которые могут быть изменены в процессе проектирования. При этом изменение одного из параметров влечет за собой и изменение других (например, изменение номинала или завода, изготовителя компонента — изменение наименования). Постоянное отслеживание соответствия всех параметров компонентов занимает много времени и не повышает производительности. Подключить базу данных можно и из примера, входящего в пакет САПР. Однако мы создадим свою базу: это достаточно просто и понятно.

Произведем следующую последовательность действий:

1. Из схемного редактора командой **Tools/ Parameter Manager**, как уже делали ранее, откроем таблицу параметров компонентов.
2. Выделим столбцы существующих параметров и скопируем их в буфер.
3. Откроем программу EXCEL и вставим значения из буфера.
4. Отсортируем строки, удалим дубликаты строк и столбцы с параметрами, которые мы не будем использовать.
5. Добавим первую строку и введем наименования столбцов.
6. Расположим столбцы в удобной для нас последовательности.
7. Добавим первый столбец в пустой столбец с именем, например Link.
8. В столбце Link введем уникальный номер для каждой строки, отражающий свойства ком-

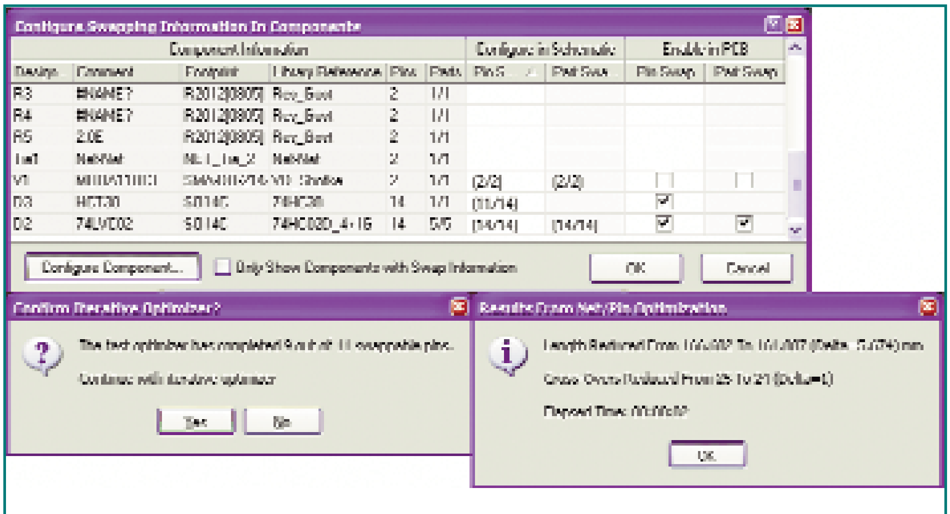

Рис. 42. Разрешение применения эквивалентности PIN и PART

Link	DataSheet	Наименование	ValueSCH	MarkPCB	Поставщик
C4532/105		Конд. TDK C4532X7R2A105M	1.0u,100V		
C3216/104		Конд. TDK C3216X7R2A104M	0.1u,100V		
C1206/104		Конд. Kemet C1206C104K5RAC	0.1u,50V		
C1206/103		Конд. Kemet C1206C103K5RAC	0.01u,50V		
C4532/156		Конд. TDK C4532X7R1H156M	15u,100V		
MSOP/5008		М-с LM5008, NC	LM5008		
SO/LVC20		М-с 74LVC02	74LVC02	*	
SO/HCT30		М-а IN74HCT30D, Интеграл	HCT30	*	
[illegible]		Дроссель TDK SLF10145T-221MR65	220uH		
R1206/3573		Рез. Vishay CRCW12063573F	357k		
R1206/2673		Рез. Vishay CRCW12062673F	267k		
R1206/8011		Рез. Vishay CRCW12068011F	[illegible]		
R1206/1001		Рез. Vishay CRCW12061001F	1k	102	
[illegible]		Рез. Vishay CRCW[illegible]	[illegible]	7R0	
DO214/MURA110		Диод ON Semi MURA110T3	MURA110T3		

Рис. 43. Готовая база

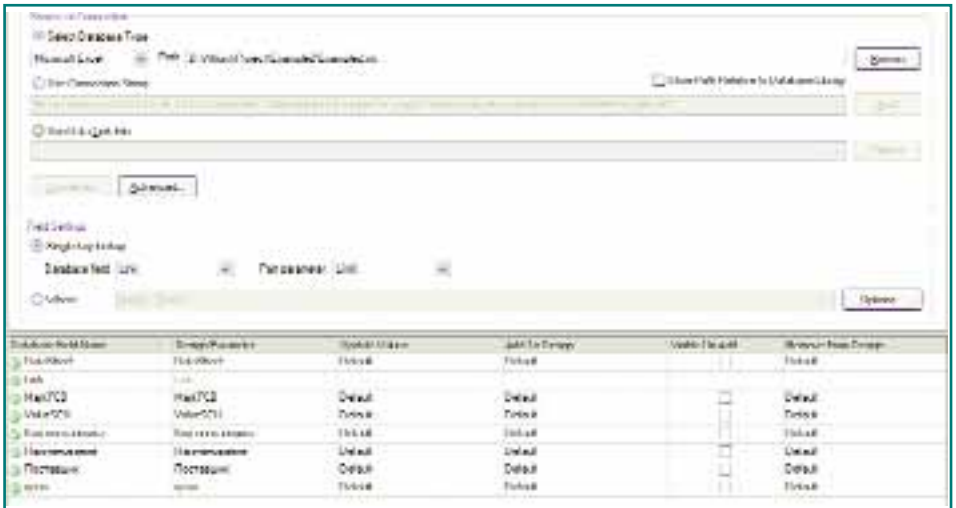

Рис. 44. Настройка базы данных

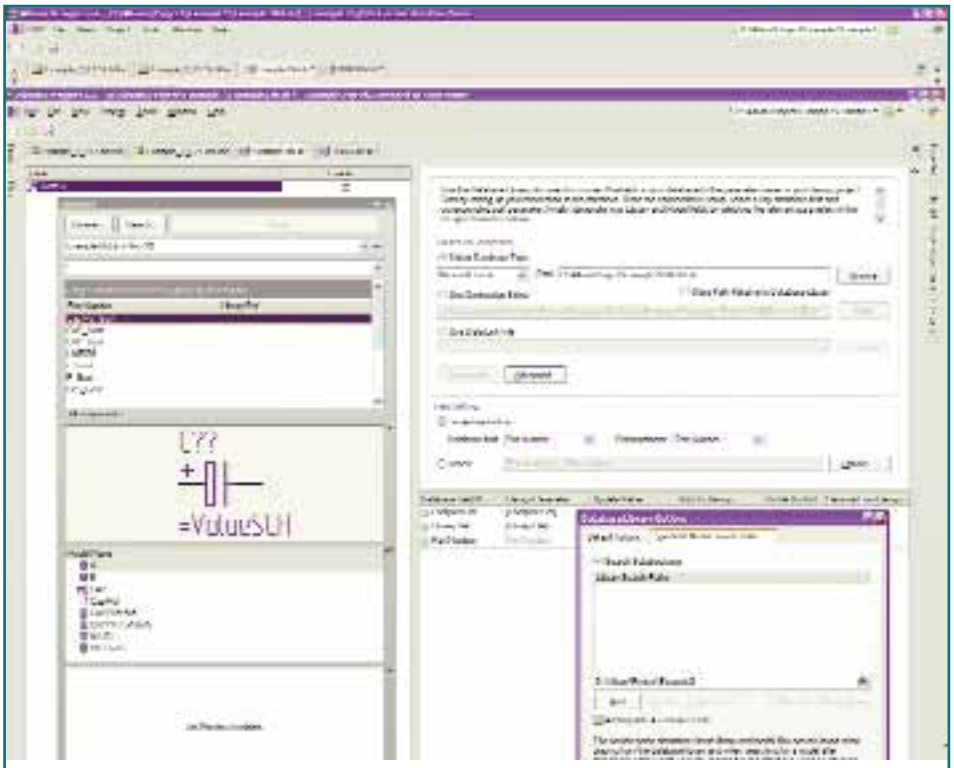

Рис. 45. Настройка базы данных DbLib

понента. Желательно ограничить его длину, так как для сверки с базой необходимы версии твердых копий схем с видимым значением этого параметра. В частности, в примере мы сформируем его название из типа корпуса и обозначения основного параметра.

9. По желанию можем добавить и заполнить и другие столбцы с информацией о поставщике, стоимость изделий, ссылки на DataSheet и так далее.
10. Сверяем таблицу с исходными данными, находим и устраняем несоответствие в параметрах.

Итак база готова, ее вид показан на рис. 43. Сохраняем его в наш проект, например, с именем Example2.

Компоненты в схеме можно связать по любым параметрам, но так как мы хотим наоборот изменять параметры в схеме, введем в ней ко всем элементам параметр **Link** и его значение в соответствии со столбцом **Link** нашей базы. С этого момента и в дальнейшем достаточно отслеживать только этот параметр и его значение в схеме, а остальные вводить или через базу или, не вводя в схему, сразу добавлять их значения в документы при генерации REPORT.

Добавим **Project/Add New To Project** к нашему проекту новый документ **DataBaseLinkDate** и сохраним с именем **Example.** В документе **Example.DBLink** отмечаем необходимые установки и производим следующие действия:

1. Создаем базу в EXCEL, в выпадающем меню Select Database Type=Microsoft Excel.
2. Нажимаем кнопку **Browse** и указываем путь к файлу базы данных.
3. Нажимаем кнопку **Connected.**
4. Выбираем параметры для проекта и баз данных, по которым будут связываться остальные параметры. В нашем примере имена параметров совпадают и имеют значение **Link**.
5. Каждому параметру, кроме **Link**, можно поставить атрибут или обновить его в схеме, или добавить в схему, если его нет, и так далее.

База данных готова.

Теперь в схемном редакторе можно использовать команду **Tools/Update from Database**, и при генерации различных **REPORT** добавлять дополнительные параметры, которые находятся в базе данных (рис. 44).

Создание базы данных библиотечных элементов

В качестве заготовки базы используем лист EXCEL файла, в котором нужно добавить столбцы:

- Part Number — для заполнения значения имени компонента в библиотеке;
- Footprint Ref — для заполнения значения имени посадочного места данного компонета;
- Library Ref — для автоматического заполнения данного параметра компонента в библиотеке;
- другие столбцы с параметрами по желанию пользователя.

С помощью команды **File/New/Library/DataBaseLibrary** создадим новую базу данных, сохраним ее под именем **Example** и подключим, как делали выше, созданный EXCEL файл. Дополнительно, через **Tools/Option,** указываем директории, где находятся библиотеки, в которых будет проводиться поиск компонентов по названиям из базы данных. Через панель **Library** проинсталлируем новую, только что сделанную нами библиотеку **Example.DbLib**.

Теперь прямо с панели **Library** мы сможем вводить и в схему, и в PCB новые компоненты, которые мы сами определили в EXCEL файл, изображение и посадочные места для которых существуют хотя бы в одной из проинсталлированных стандартных библиотек.

Пример заполнения параметров и свойств базы данных и вид панели **Library** при выборе компонента из библиотеки представлен на рис. 45.

Теперь добавление нового библиотечного компонента, не имеющего отдельного изображения на схеме и специфического посадочного места, производится путем записи в базе, даже без открытия пакета САПР.

Заключение

Автор надеется, что его статья позволит снять целый ряд неизбежных вопросов, возникающих при освоении данного пакета САПР, окажет помощь пользователям в быстром переходе от изучения к практической реализации первых собственных проектов и сократит время до получения полностью завершенных работ от разработки до получения печатных плат.