Altium Designer: преимущества и недостатки организации библиотеки в виде базы данных

Алексей Сабунин (Москва)

При соответствующем подходе к организации библиотек компонентов в рамках крупных предприятий можно решить множество насущных проблем именно на стадии создания базы данных, если учесть все необходимые аспекты, о которых рассказывается в статье.

Введение

В отличие от аналогичных систем проектирования, программа Altium Designer позволяет организовать библиотеки как минимум тремя различными способами. Поэтому первый вопрос, который задают специалисты на предприятиях при внедрении Altium Designer: как правильно организовать библиотеки? Безусловно, каждый из предлагаемых вариантов имеет преимущества и недостатки. В рамках крупных предприятий оптимальным решением является организация библиотеки в виде базы данных (БД), но даже при таком подходе остаются открытыми некоторые вопросы реализации такой базы.

В данной статье мы не будем затрагивать создание библиотеки в виде БД, поскольку это описано в статье [3] и в оригинальной документации [4]. Во-первых, мы проведём сравнительный анализ различных способов организации библиотек в Altium Designer, чтобы ответить на вопрос: какой метод использовать на предприятии? Во-вторых, рассмотрим возможные проблемы при создании библиотеки в виде БД, на которые следует обратить первостепенное внимание на начальных этапах такой работы.

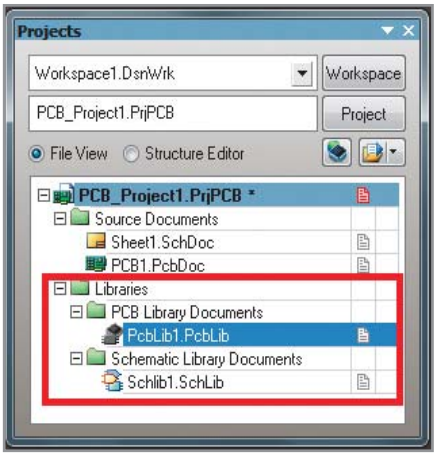

Рис. 1. Библиотеки проекта

Способы организации библиотек

Как было сказано ранее [1], библиотеки можно организовать в Altium Designer как минимум тремя способами, к которым сегодня можно добавить четвёртый способ – Vault.

Первый способ – *библиотеки проекта*. В данном случае условные графические обозначения (символы) хранятся в библиотеках *.SchLib, а посадочные места (footprint) – в библиотеке *.PcbLib (см. рис. 1). При этом схема строится из символов, к ним вручную привязывается посадочное место (это может быть сделано чуть раньше, в библиотеке символов), после чего схема передаётся в редактор плат. Поэтому библиотека создаётся индивидуально для каждого проекта и состоит из двух файлов. Всю ответственность несёт разработчик проекта, что практически неосуществимо на больших предприятиях. Такой подход можно использовать в том случае, если весь проект ведёт один разработчик или – как временное решение – когда существует централизованная база данных, но в ней отсутствует требуемый компонент.

Второй способ – *интегрированные библиотеки*. Для создания такой библиотеки необходим отдельный проект, т.н. проект библиотеки *.LibPkg (см. рис. 2). Внутри этого проекта хранятся библиотеки с символами, посадочными местами и файлы других моделей (Spice, IBIS). Каждому компоненту (символу, т.к. каждый объект библиотеки символов есть компонент) можно сопоставить посадочное место, модель и добавить набор параметров, отражающих его характеристики. При использовании такого метода останется лишь расставить компоненты по схеме, соединить их, и можно передавать проект в редактор плат. Облегчается генерация BOM (перечня элементов), спецификации и других сопутствующих документов. У интегрированной библиотеки есть определённые преимущества:

- компактность. В итоге у пользователя будет один файл (*.IntLib), внутри которого будут храниться все используемые символы и посадочные места. При необходимости переноса такой библиотеки с одного компьютера на другой достаточно передать один файл;
- архивация. При создании файла интегрированной библиотеки происходит упаковка проекта библиотеки, и размер файла уменьшается в несколько раз по отношению к содержимому проекта;
- простота. Такой метод наиболее близок тем пользователям, которые работали с программой P-CAD, где всё содержимое библиотеки хранилось в едином файле (*.lib).

Главный недостаток заключается в том, что символ и компонент в интегрированной библиотеке – это единая сущность, т.е. у каждого компонента свой символ и свой набор атрибутов. Исходя из этого, существенно увеличивается объём библиотеки и отсутствует необходимая гибкость в её использовании. Например, если в базе хранится 1000 резисторов, это означает хранение такого же количества одинаковых символов, и при необходимости внесения изменения в графику символа необходимо будет выполнить одинаковую процедуру с каждым компонентом.

Тем не менее, такой способ организации библиотек на данный момент является самым распространённым на предприятиях, которые внедряют Altium Designer.

Третий способ – *библиотека в виде базы данных*. При такой организации основную часть библиотеки представляет БД, в которой содержатся записи о компонентах, где для каждой записи указан применяемый символ, посадочное место и другие модели, а также набор атрибутов (см. рис. 3). Составные части компонента хранятся отдельно, при этом желательно разделить их на отдельные каталоги

(символы, посадочные места, модели и т.д.).

Библиотека в виде базы данных может быть в двух форматах – Database Library и SVN Database Library, которые отличаются только способом хранения символов и посадочных мест. В первом случае все символы хранятся в одной библиотеке, а все посадочные места – в другой. Во втором случае для каждого символа и посадочного места создаётся индивидуальная библиотека. Такой подход обеспечивает организацию контроля версий: изменился файл библиотеки, значит, изменился символ или посадочное место, которое хранится в данном файле. Контроль версий влечёт за собой хранение библиотек на сервере с сохранением записей о всех манипуляциях с компонентами.

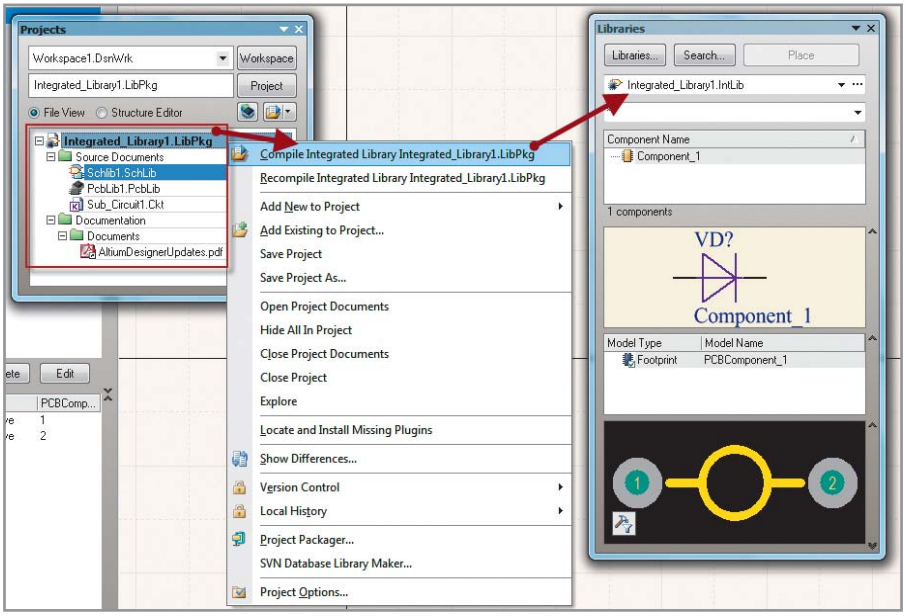

Рис. 2. Создание интегрированной библиотеки

Использование БД в качестве основы библиотеки Altium Designer даёт пользователю целый ряд преимуществ. Главное из них заключается в том, что в библиотеках *.SchLib теперь хранятся именно символы, которые могут быть многократно использованы в различных компонентах. Компонент создаётся на стадии добавления записи в таблицу БД, где путём копирования записей можно достаточно быстро увеличивать объём базы. Кроме того, появляется возможность интегрировать БД в систему складского и бухгалтерского учёта или с PDM-системами.

В качестве недостатка можно отметить увеличение трудоёмкости создания библиотеки. Для ведения БД необходимо выделять отдельного, «специально обученного» человека, который будет нести за неё ответственность. Ещё один недостаток БД – в ней нельзя заполнить таблицу соответствия выводов символа и посадочного места (как в P-CAD > Library Executive), такое соответствие определяется автоматически по ранее заданным позиционным

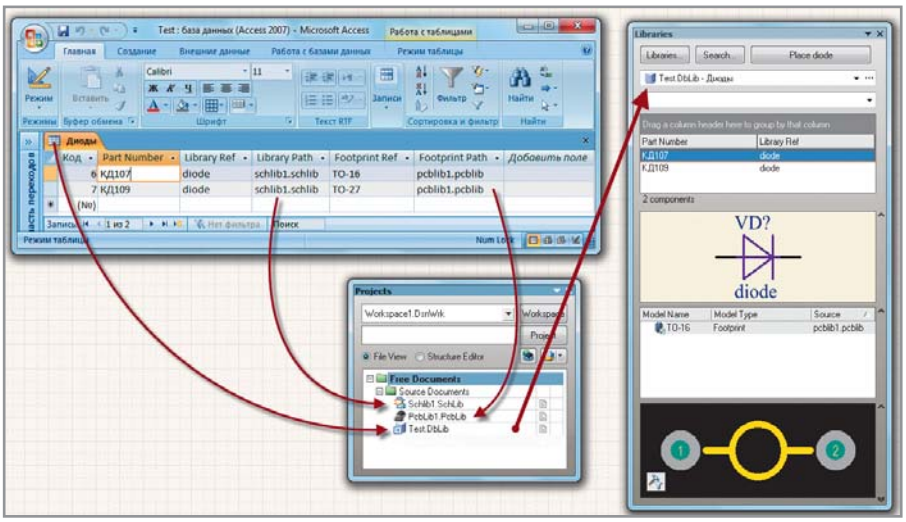

Рис. 3. Библиотека в виде базы данных

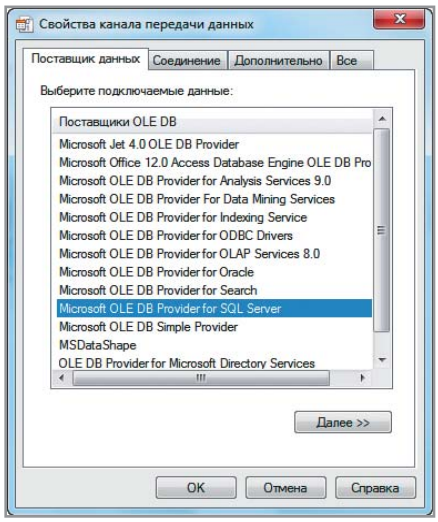

Рис. 4. Подключение к базе данных

обозначениям (Designator) выводов. Это особенно непривычно для тех пользователей, которые работали в программе P-CAD, но такой подход фактически является стандартом во всех «серьёзных» САПР (Allegro, OrCAD, Expedition).

Структура БД

Итак, мы пришли к выводу, что на больших предприятиях библиотеку электрорадиоизделий для Altium Designer лучше вести в виде базы данных, чтобы обеспечить максимальную применяемость одинаковых объектов (символов и посадочных мест) и возможность синхронизации с другими БД (PDM, склад и др.). Преимущества такого подхода очевидны, но чтобы их максимально использовать, необходимо правильно организовать БД. Прежде чем это сделать, необходимо ответить на два вопроса: во-первых, сколько таблиц будет иметь база данных и, во-вторых, сколько атрибутов будет содержать каждая таблица базы данных? От того, насколько грамотно будут проработаны эти вопросы, в конечном итоге будет зависеть удобство использования и гибкость библиотеки. Перед этим предстоит выбрать СУБД, на основе которой всё это будет реализовано.

При создании файла *.DBLib, который является интерфейсом между Altium Designer и внешней БД, предлагается использовать в качестве основы базу в формате MS Access или MS Excel. Эти системы хорошо известны большинству пользователей и удобны при освоении работы с библиотеками в виде БД в программе Altium Designer. Но обе эти программы имеют проблемы с многопользовательским доступом и некоторые другие нюансы, которые не позволяют применять их в промышленных масштабах. Поэтому на предприятиях для реализации БД используют более мощные СУБД, а в Altium Designer при желании можно подключить практически любую БД через драйвер ODBC, например, MS SQL Server или даже текстовые файлы в формате CSV (см. рис. 4).

Сколько должно быть таблиц в БД и какие поля должна иметь каждая таблица? На самом деле эти вопросы во многом взаимосвязаны. Так, например, отвечая на вопрос о том, сколько таблиц должна содержать база данных, необходимо определить задачи деления БД на таблицы. Можно создать одну таблицу, «слить» в неё все разнородные компоненты и, используя запросы, извлекать нужные компоненты. Кстати, на многих предприятиях так и делают, несмотря на то что при этом существенно возрастает трудоёмкость работы с базой через СУБД.

Следует отметить, что такие предприятия чаще всего не используют инструменты СУБД для наполнения базы, а создают для этого пользовательские программы. Предполагается, что разделение на таблицы необходимо для того, чтобы компоненты, содержащие разный набор атрибутов, оказались в разных таблицах, что упростило бы работу с ними. Можно сделать одну таблицу и всем компонентам задать одинаковый список атрибутов (если ограничиться неким минимумом атрибутов), либо разделить компоненты по семействам с разными атрибутами и хранить их в различных таблицах.

Следствием разделения базы на таблицы по типам компонентов будет (точнее, может быть) разделение символов, посадочных мест и моделей по разным библиотекам, что довольно удобно для работы. На практике компоненты обычно делят на таблицы по указанному ниже принципу.

По разделам:

- Resistors (резисторы);
- Capacitors (конденсаторы);
- Inductors (катушки индуктивности и трансформаторы);
- Integrated circuits (микросхемы);
- Semiconductors (полупроводниковые приборы);
- Connectors (разъёмы, соединители);
- Miscellaneous (прочее: реле, ключи и т.д.).

По производителю:

- отечественные;
- импортные (по фирме-производителю).

Очевидно, такое разделение не претендует на полноту и оригинальность, но является вполне рабочим и используется почти повсеместно. При таком подходе БД будет содержать отдельные таблицы для отечественных и импортных резисторов (резисторы всех импортных производителей в одной таблице, т.к. они привязаны к единым символам и посадочным местам). Отечественные микросхемы сводятся в единую таблицу, а импортные делятся по таблицам, в зависимости от производителя. Это связано с тем, что у разных производителей по-разному именуются посадочные места в документации (рекомендуется придерживаться этих наименований) и, к тому же, компоненты каждого производителя будут в большей степени содержать уникальный набор символов. Пример такой организации можно найти в документе «Комплект библиотек для Altium Designer» [5].

Параметры компонентов БД

В каждой из таблиц БД будут содержаться параметры компонента, которые можно разделить на три группы: идентификатор, зарезервированные (системные) параметры и параметры общего назначения (пользовательские).

Идентификатор – это одно (простой идентификатор) или несколько полей (сложный идентификатор), по которым Altium находит и идентифицирует компонент. Идентификатор компонента определяется в файле DbLib (см. рис. 5), где:

- Single key lookup – простой идентификатор;
- Where – сложный идентификатор.

Следует отметить, что со сложными идентификаторами Altium работает неустойчиво, и поэтому использовать их не рекомендуется. Идентификатор компонента является обязательным, и он должен быть уникальным. Самое простое решение – автоинкрементируемое числовое поле (индекс), которое всегда является уникальным и не требует определения при добавлении нового компонента. Недостаток такого подхода заключается в том, что индексы не содержат в себе никакой информации о самом компоненте.

Более удачным решением является идентификация компонента по сочетанию двух параметров – «наименование производителя» и «наименование компонента» для импортных компонентов, и запись по ТУ для отечественных. По сути это формат записи компонента в спецификации. Можно сделать «автособираемый» атрибут из некоторого числа параметров компонента, но, как упоминалось выше, со сложными идентификаторами Altium работает некорректно. Проблема была решена вводом в таблицу уникального поля Part Number, которое хранит в себе запись для спецификации. Это поле заполняется пользователем и используется как идентификатор компонента в базе.

Зарезервированные параметры – это параметры, при нахождении которых Altium заполняет форму свойств компонента, например, символ, посадочное место, модель, тип и т.д. Эти параметры показаны на рисунке 6 (1, 2). Полный список зарезервированных параметров можно найти в оригинальной документации [4]. Обязательными среди них является только Library Ref – название символа УГО – и Library Path – местоположение символа. Остальные зарезервированные параметры добавляются по мере необходимости.

Рис. 5. Определение идентификатора компонента

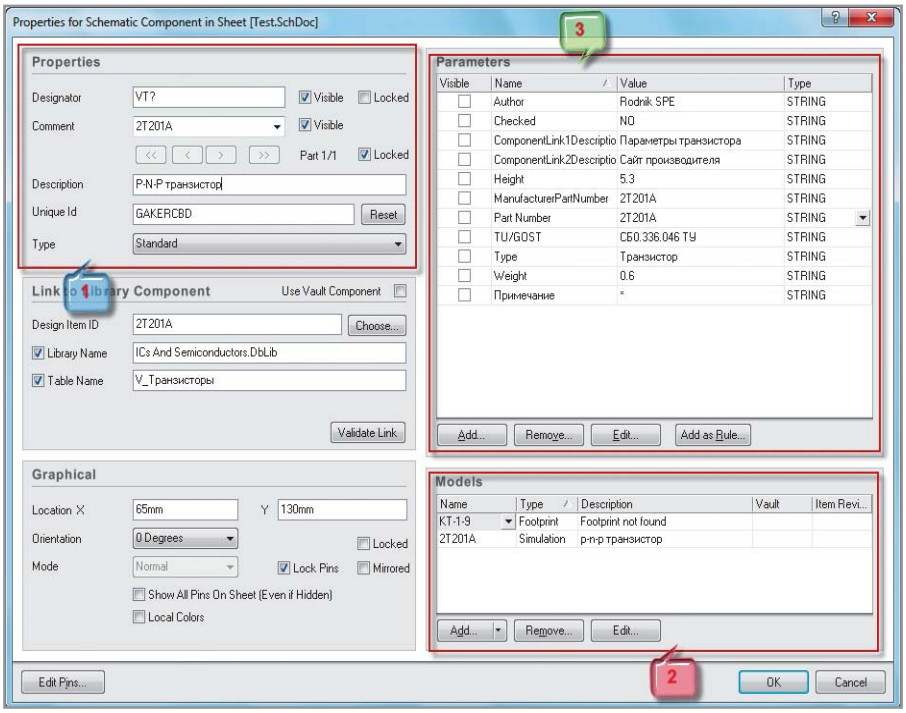

Рис. 6. Параметры компонента

Перечень пользовательских параметров (3, см. рис. 6) определяется на каждом предприятии индивидуально, но в последнее время практически все приходят к одинаковому набору.

Этот набор параметров должен обеспечить решение двух задач: 1) формирование запроса на поиск компонента в базе данных и 2) оформление текстовой конструкторской документации (спецификации, перечня и др.). Рекомендуется брать за основу карточку компонентов ведущих мировых поставщиков, например, Digi-Key (http://www.digikey.com), у которого в БД хранится наиболее подробное описание компонентов.

Заключение

При правильном подходе к организации библиотек в рамках крупных предприятий можно решить множество насущных проблем именно на стадии создания базы данных, если учесть все необходимые аспекты, которые были рассмотрены выше. Маршрут разработки базы библиотек компонентов необходимо рассматривать как самостоятельный процесс, и ему надо уделять большее внимание, чем процессу разработки схем или печатных плат. В Altium Designer библиотеки следует вести именно в формате базы данных, работу с которыми поддерживают и другие известные САПР (OrCAD, Allegro, Expedition). При необходимости использования этих систем, изменения затронут лишь символ и посадочное место, которые следует создать в формате данной САПР, а сама БД останется неизменной.

Литература

1. *Сабунин А.Е.* Altium Designer. Новые решения в проектировании электронных устройств. Солон-Пресс, 2009.
2. *Сабунин А.Е.* Altium Designer Summer 08 – разработка библиотек и моделей компонентов. Современная электроника. 2008. № 6.
3. *Пранович В.* Altium Designer 7. Создание библиотеки на основе базы данных. Технологии в электронной промышленности. 2008. № 5.
4. http://www.altium.com/files/Altiumdesigner6/LearningGuides/AP0133. Using Components Directly from Your Company Database.pdf.
5. http://www.rodnik.ru/AltiumLibrary.