МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра космического приборостроения и систем связи

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«15» 12 2017 г.

Структура пакета ORCAD и разработка электронного компонента принципиальной схемы

Методические указания по выполнению лабораторной работы
для студентов, обучающихся по направлению подготовки 11.03.03

Курск 2017

УДК 681.5

Составитель А.А. Гримов

Рецензент

Доктор технических наук, профессор А.Ф. Рыбочкин

Структура пакета ORCAD и разработка электронного компонента принципиальной схемы: методические указания по выполнению лабораторной работы по дисциплине «Информационные технологии конструирования электронных средств» / Юго-Зап. гос. ун-т.; сост.: А. А. Гримов. Курск, 2017. 23 с.: ил. 9.

Содержатся теоретические сведения, касающиеся разработки электронных компонентов принципиальных схем в среде специализированного графического редактора ORCAD CAPTURE.

Указывается порядок выполнения лабораторной работы.

Текст печатается в авторской редакции

Подписано в печать . Формат 60×84 1/16.
Усл. печ. л. ___. Уч.-изд. л. ___. Тираж 100 экз. Заказ . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение основных характеристик пакета **OrCAD** и разработка электронного компонента принципиальной схемы с помощью графического редактора **Part Editor**.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В теоретической части рассматриваются структура, свойства и основные характеристики пакета **OrCAD**, а также графический редактор Part Editor и алгоритм разработки электронных компонентов принципиальных схем. Дается пример разработки символа электронного компонента.

Основные параметры, свойства и структура пакета OrCAD

Система автоматизированного проектирования **OrCAD** является одним из лидеров на рынке инструментальных средств проектирования электронной аппаратуры.

Пакет **OrCAD** включает в себя программы (рис. 1):

- создания электронных схем **OrCAD Capture**;
- моделирования работы цифровых схем **OrCAD Simulate** (в составе программы OrCAD Capture);
- моделирования работы аналоговых и цифровых схем **OrCAD Pspice A/D**;
- проектирования печатных плат **PCB Layout**.
- подготовки и вывода на печать конструкторской документации и производства печатных плат.

Программа проектирования электронных схем **OrCAD Capture** состоит из:

- *входного редактора*, обеспечивающего создание новых и загрузку существующих проектов электронных схем, создание новых и подключение существующих библиотек электронных компонен-тов (далее ЭК), а также переход в другие редакторы;
- *менеджера проектов,* обеспечивающего полный процесс

разработки электронных схем и подготовки необходимой документации совместно с моделированием работы *электронных схем* и взаимосвязь между редакторами, входящих в **OrCAD Capture**;

– *графического редактора электронных компонентов* **Part Editor**;
– *графического редактора электронных схем* **Capture**;
– *редактора текста* **Text Editor**;

редактора свойств электронных компонентов **Property Editor**;

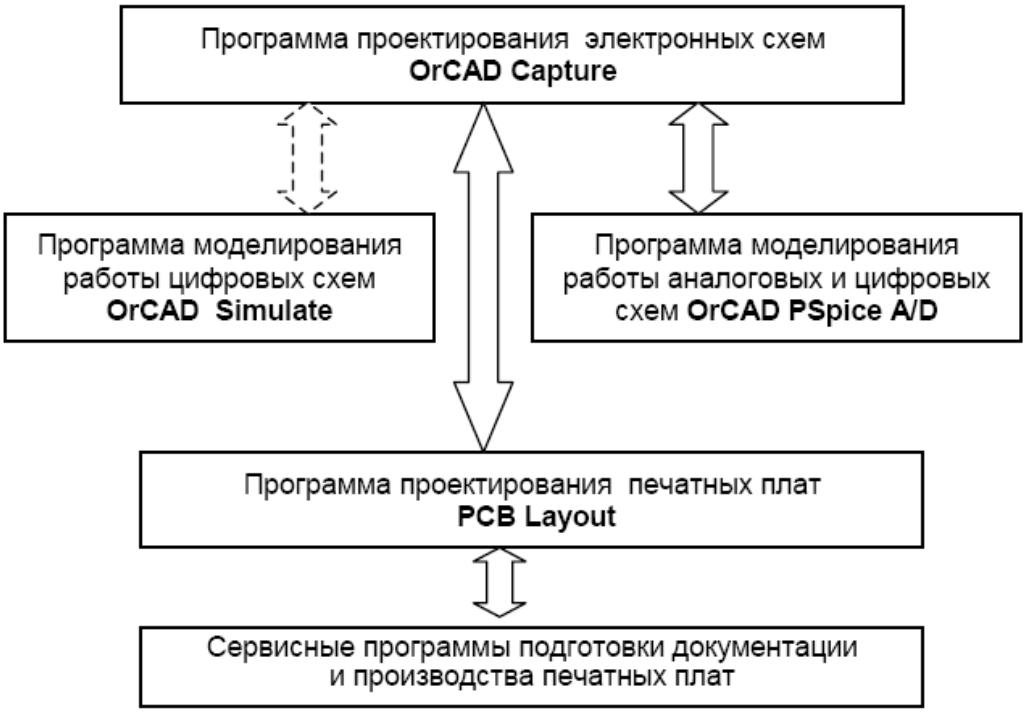


Рис. 1. *Структурная схема пакета* ***OrCAD***

Доступ к программам этого пакета осуществляется из стартового меню **Windows**. Щёлкнем мышкой на кнопке **Пуск** и выберем команду **Программы**. В появившемся списке приложений найдём имя нужного пакета **OrCAD_9.1 LITE** и затем в подменю – команду запуска программы **Capture.**

После загрузки редактора на экране в верхней части дисплея появится ***строка команд главного меню*** (далее - ***ГМ***) с

выпадающими ***подменю***, а немного ниже – ***панель инструментов,*** на которой размещены кнопки наиболее часто используемых команд ГМ. Эту панель при желании можно отключить командой главного меню **View/Toolbar.** В нижней части экрана расположена ***статусная строка***, куда выводится информация о командах главного меню, состоянии сервисных программ и проекта.

Проектирование новой схемы начинается с создания нового проекта. Введем команду **File/New/Project** или щёлкнем по кнопке **Create document**, расположенной на панели инструментов. На экране появится диалоговая панель **New Project**, в которой надо задать имя проекта, выбрать тип проекта и определить, где он будет располагаться на жёстком диске. Аналогично загружаются готовые проекты с помощью команды **File/Open/Project,** которая открывает окно каталогов, в которых, например, в **Sample** хранятся рабочие проекты. Также можно создать новую (или открыть уже существующую) библиотеку с помощью команды **File/New/ Library** или **File/ Open/ Library** и задать ей уникальное имя, например **my_lib.olb**.

Кроме того, с помощью команд **File/New/Text** и **File/Open/Text** можно открыть текстовый редактор **Text Editor** для создания и редактирования текстовых.

После выполнения всех перечисленных выше операций по созданию или загрузке готовых проектов или библиотек, слева на экране дисплея появится ***менеджер проекта*** с введенным названием проекта или, в данном случае, библиотеки ***«my_lib.olb»,*** щёлкнув ***правой кнопкой мыши*** по имени которой, выберем ***из выпадающего меню*** команду **New Part** для создания нового электронного компонента. При этом произойдет переход из *входного редактора* **Capture** в *редактор проектирования электронных компонентов* **Part Editor**. На экране дисплея появится графическая оболочка редактора **Part Editor**, которая отличается от *входного редактора* **Capture** только командами главного меню. Эту же операцию в ***менеджере проекта*** можно осуществить с помощью команды главного меню **Design/New Part**.

Менеджер проекта

Рассмотрим структуру и особенности работы *менеджера проекта.* Экран редакторов **Part Editor** и **Capture** содержит окно *менеджера проекта (как правило,*слева) и окно *области рисования, в которой находится* электронная схема (справа).

Впрочем, таким будет экран, если в редактор **Capture** загружен какой-либо проект. С помощью этого редактора создаётся принципиальная схема проектируемого устройства.

Менеджер проекта работает в двух режимах:

- файловая структура проекта – **File**;
- иерархическая структура проекта – **Hierarchy**.

В *первом случае* файловая структура проекта состоит из следующих разделов:

• **Design Resource**, включающего в себя файлы проектов, отдельных страниц электронных схем, библиотеку электронных компонентов (ЭК), входящих в проект – **Design Cache**, подключенные к проекту библиотеки;

• **Outputs,** включающий в себя результаты проектирования;

• **Pspice Resource** (редактор **Pspice A/D**). **Pspice Resource** включает в себя подразделы **Include Files**, **Model Libraries**, **Simulation Profiles** и **Stimulus Profiles**.

Во *втором случае* иерархическая структура проекта состоит из: иерархических блоков (**Hierarchical block**), принципиальная схема каждого их которых расположена на том же уровне, что и основная схема.

Обратим внимание на то, что *состав* команд главного меню и их подменю (*менеджера проекта*, *редакторов* **Part Editor** и **Capture**) зависит от того, какое из названных окон активно.

Кроме того, на *менеджера проекта* возлагается ещё одна весьма важная функция: он служит также управляющей оболочкой, из которой можно подключать к проекту другие редакторы и программы.

Понятно, что запустить редакторы **Capture** или **Part Editor** можно и другими способами. Например, если в *менеджере проекта* уже имеются файлы, созданные этими редакторами

(файлы с расширением *.**opj**, *.**dsn** или *.**olb** и т.д.), то достаточно выделить любой из них и дважды щёлкнуть мышью на его имени. На рис. 1.6 представлены *команды главного меню (ГМ) с командами подменю* ***менеджера проекта***, который автоматически подключается при создании проекта или библиотек.

Особенностью *ГМ менеджера проекта* является появление новых команд **Design**, **Tools**, **Accessories** и **Reports**.

Как отмечалось выше, с помощью команды **Design** (команды подменю типа **Design/New Part**, **Design/New Schematic** и т.д.) создаются новые электронные компоненты и схемы в графическом и текстовом представлении.

Команда **Tools** обеспечивает автоматическую нумерацию ЭК на принципиальной схеме (команда подменю **Tools/ Annotate**), проверку правил выполнения электронных схем (команда подменю **Tools/Design Rules Check**), создание списка соединений и ЭК принципиальной схемы или **Net**-файл для перехода в программы моделирования работы электронных схем или проектирования печатных плат с помощью команды подменю **Tools/Create Netlist**. Спецификация ЭК для принципиальной схемы создается автоматически с помощью команды подменю **Tools/Bill of Materials**.

Команда **Accessories** предназначена для связи проектов **OrCAD Capture** с пакетом программ **Allegro**.

Необходимые отчеты о проектах составляются с помощью ко-манды **Reports**.

Главное меню данного редактора **Part Editor** отличается от ГМ *входного редактора* появлением новой команды **Place**, позволяющей создавать и редактировать электронные компоненты принципиальных схем.

При включении команды **Place** появляется подменю команд, которые продублированы в инструментальной панели **Tool Palette**

По умолчанию панель **Tool Palette** располагается вертикально в правой части экрана. Так как инструменты этой панели очень важны, прокомментируем их более подробно.

Наиболее часто используемые кнопки ***инструментальной панели***:

– Кнопка **Select** предназначена для переключения курсора

мыши в режим выбора (выделения) объектов схемы (элементов, це-пей, имён и т.п.).
- Кнопка **Pin** позволяет размещать выводы компонентов.
- Кнопка **Pin Array** позволяет размещать массив выводов компонентов.
- Кнопка **IEEE Symbol** обеспечивает введение обозначений ЭК в американском стандарте.
- Кнопка **Line** позволяет рисовать линии.
- Кнопка **Polyline** позволяет рисовать ломаные линии.
- Кнопка **Arc** позволяет рисовать дуги.
- Кнопка **Elipse** позволяет рисовать эллипсы.
- Кнопка **Rectangle** позволяет рисовать прямоугольники.
- Кнопка **Text** позволяет вводить текст.

Названных кнопок вполне достаточно, чтобы создавать схемы электронных компонентов, поэтому остальные команды оставим пока без внимания. При этом команды **Help/Learning Capture** и **Help/Topics** предназначены для самостоятельного изучения редак-тора **Capture** и нахождения ответов на возникающие вопросы по проектированию электронных схем.

Методика и особенности проектирования символов электронных компонентов

Как создавать символы электронных компонентов

На принципиальной схеме электронный компонент представляется в виде условного графического обозначения (**УГО**). Его при-нято называть символом. Символы электронных компонентов объ-единяются в библиотеки, имеющие расширение ***.olb**. Символы в свою очередь делятся на *основные* и *вспомогательные* (рис. 2).

Вспомогательные символы представляют собой небольшую группу, куда входят соединители страниц (**Off-Page Connector**), порты иерархических блоков (**Hierarchical Port**), «угловые штампы» (**Title Block**) и символы подключения цепей питания (**Power**). Они создаются командой **New Symbol** из всплывающего меню или командой **Design/New Symbol** из выпадающего меню.

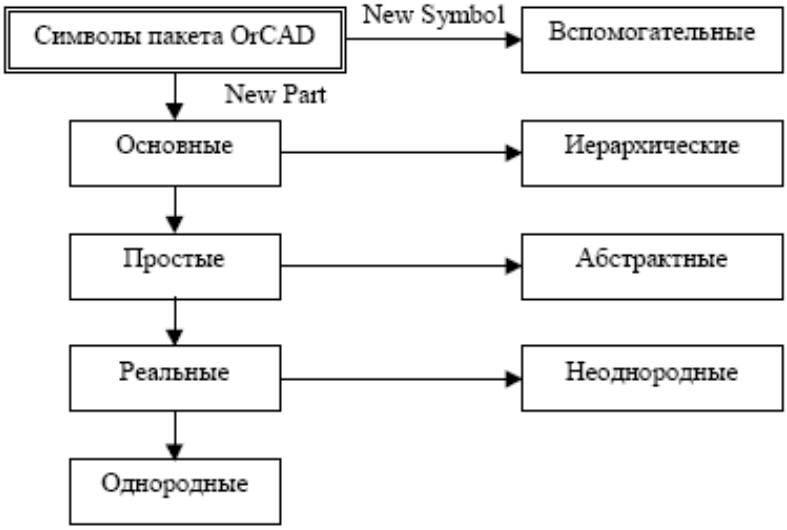


Рис. 2. *Разновидности схемных описаний* ***символов*** *электронных компонентов в пакете* ***OrCAD***

Подавляющее большинство ЭК относится к группе *основных* символов. Процесс их проектирования инициируется другой командой – **New Part**. Слово **Part** (дословно – часть, корпус) может ввести пользователей в заблуждение. В пакете **OrCAD** команда **Part** – это вовсе не корпус, а графическое изображение компонента на схеме. Однако, если компонент состоит из нескольких секций, то изображение любой из них будет называться другим словом – **Symbol**. Таким образом, перевод слова **Symbol** зависит от контекста: это может быть *вспомогательный* символ или графический *образ одной секции* основного компонента.

Основные символы подразделяются на *простые* и *иерархические*. ***Простые*** символы называют структурными *примитивами* и в ходе проектирования они не подлежат какому-либо расчленению (детализации). ***Примитивы*** представляют собой объекты с известным поведением. Другими словами, они всегда имеют функциональное описание, часто в виде встроенных в систему поведенческих моделей. В пакете **OrCAD** – это **VHDL**- или **SPICE**-модели ЭК.

Иерархические символы в САПР **OrCAD** реализованы в виде иерархических блоков (**Hierarchical Block**) или непримитивных

символов (**Nonprimitive Part**). *Иерархические символы* имеют два описания – внешнее и внутреннее.

При понижении уровня описания такого символа, он превращается в структуру, показывая «спрятанную» в нём **подсхему** или **VHDL-команды** в виде программ. Внутреннее описание в большинстве случаев – это подчинённая схема, которую часто называют схемой замещения или эквивалентной схемой.

Среди *простых символов* можно, в свою очередь, выделить две разновидности:

- символы **реальных** электронных компонентов;
- символы **абстрактных** электронных компонентов. Отличительным признаком ***реального*** *электронного компонента*, например микросхемы **555LA3**, является наличие упаковочной информации в его схемном описании, с указанием контактов питания и земли, и ссылка на конструкторское описание (тип корпуса, например **DIP14** или **SO14**).

Основное отличие ***абстрактных*** *электронных компонентов* заключается в том, что для них не указывается, а, возможно, и не существует конкретная физическая реализация. Например, мы можем создать символ компонента NAND2 (логический элемент 2И-НЕ), не утруждая себя описанием упаковочной информации и ссылками на тип корпуса. Никто не запретит вам нарисовать двухвходовой

мультиплексор **MUX2**, описать его функцию и использовать затем в проекте, хотя в действительности такой микросхемы нет.

Из-за отсутствия данных о физической реализации, абстрактные компоненты не позволяют довести проект до реального воплощения. Тем не менее, они широко применяются на первых этапах функционального проектирования и при построении иерархических блоков.

И, наконец, ***реальные*** *символы* делятся на ***однородные*** (**Homogeneous**) и ***неоднородные*** (**Heterogeneous**). Первые описывают компоненты, состоящие из секций (элементов) одного типа (например, 555LA3 содержит четыре *одинаковых* элемента 2И-НЕ), а вторые – из секций разного типа (например, 555LR11

включает два *различных* элемента – 2-2И-2ИЛИ-НЕ и 3-3И-2ИЛИ-НЕ).

Проектирование абстрактных электронных компонентов

Начнём с самого простого – с процесса создания абстрактных символов, которым не требуется задавать упаковочную информацию и определять физическую реализацию. Нет необходимости вводить контакты питания и земли.

Запустим графический редактор **Part Editor** и выясним, устраивают ли нас его параметры, заданные по умолчанию.

Вход в редактор ***Part Editor***. Создадим новую (или откроем уже существующую) библиотеку, в которую будем помещать проектируемые символы (команда **File/New/Library** или **File/Open/Library**) и дадим ей уникальное имя, например **my_lib.olb**.

Установка шаблонов проектирования ЭК. Исполним команду **Options/Design Template**. После того как откроется диалоговая панель с одноимённым названием, выберем закладку **Page Size**. Уста-новим метрическую систему единиц (Millimeters), стандартный размер страницы **A4** и шаг сетки (**Pin-to-Pin Spacing**), равным 2.50 millimeters. Нажмём кнопку **OK**. Эта информация запишется в файл **Capture.ini** (в раздел [**Design Template**]) и все новые проекты будут получать по умолчанию сделанные установки.

Создание электронного компонента. Вновь щёлкнем *правой кнопкой мыши* на имени библиотеки и выберем в появившемся меню команду **New Part** – создать графический образ (УГО) нового компонента. Напомним, что команда **New Symbol** нам не подходит – она используется только для проектирования *вспомогательных* символов, например, «углового штампа» или иерархических портов. Для простоты выберем в качестве объекта проектирования двухвходовой вентиль 2И-НЕ и назовём его NAND2. После исполнения команды **New Part** на экране появится диалоговая панель **New Part Properties** (рис. 3), где нам предстоит

немного поработать. Прежде всего, в поле **Name**: надо ввести имя проектируемого символа NAND2.

Затем в поле **Part Reference Prefix**: заменим префикс **U** на **DD**. Дело в том, что по отечественным стандартам позиционные обозначения цифровых элементов начинаются именно с этих букв, на-пример DD1.1, DD3.6, DD18 и т.д.

В поле **PCB Footprint** задаётся тип корпуса, в который помещается реальный компонент. Для абстрактного символа такая информация не имеет смысла, поэтому оставим его пустым.

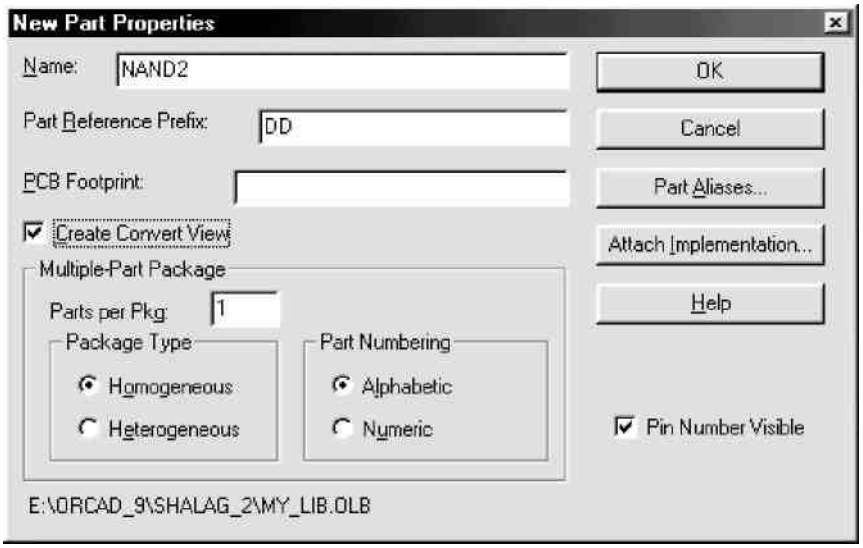


Рис. 3. *Диалоговая панель для задания основных параметров символа ЭК*

Установим флажок **Create Convert View**, чтобы показать, что для проектируемого элемента 2И-НЕ существует логически эквивалентное обозначение (рис. 4).

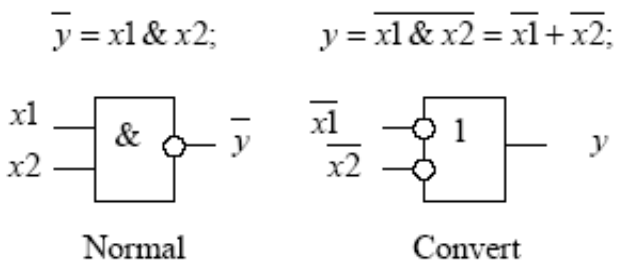


Рис. 4. *Две логически эквивалентные формы изображения элемента 2И-НЕ*

Скажем больше, мы не только отмечаем этот отрадный факт, но и собираемся нарисовать оба изображения (нормальное Normal и конвертированное **Convert**).

В следующем небольшом окне **Parts per Pkg** требуется установить число элементов, размещаемых (упаковываемых) в корпусе микросхемы. Оставим здесь цифру 1, заданную умолчанием. Для абстрактного символа эта информация не несёт реального смысла.

В разделах **Package Type** (тип упаковки – однородные или неоднородные символы) и **Part Numbering** (способ нумерации символов в упаковке – буквенный или цифровой) оставим всё как есть. В итоге диалоговая панель **New Part Properties** должна выглядеть так, как показано на рис. 3.

Если вам захочется в последствии вернуться к рассмотренной панели, то придётся воспользоваться командой **Options/Package Properties**, хотя, судя по названию, она совсем не подходит для этих целей.

И ещё один полезный совет. Первые эксперименты, как правило, бывают неудачными и нередко появляется желание удалить сделанное и начать всё заново. *Если Вы захотите удалить свой элемент, то предварительно надо закрыть окно с УГО проектируемого компонента*, и только тогда станет доступной команда **Design/Delete** или клавиша **DEL**. После нажатия на кнопку **OK** вы увидите рабочее окно графического редактора **Part and Symbol Editor**, на котором пунктиром изображён шаблон будущего символа Его габариты всегда можно изменить «буксировкой» углов. Не трудно сообразить, что необходимые размеры символа определяются числом входных (или выходных) контактов. Поэтому начинать проектирование лучше всего с размещения именно этих объектов.

В правой (по умолчанию) части рабочего окна находится палитра инструментов **Tool Palette**, используемых для создания символа. Конечно, все они могут быть вызваны и из выпадающего меню **Place.**

Щёлкнем на иконке **Place pin**, показанной справа, или активизируем команду с тем же названием. На экране появится

диалоговая панель для задания параметров контакта. Введём его имя IN1, номер 1, форму вывода **Line** и тип вывода **Input** (рис. 5).

Нажмём кнопку **OK** и поместим первый контакт слева от пунктирного прямоугольника.

Закончив размещать выводы, дважды щёлкнем на втором контакте. Убедимся, что там всё правильно – редактор сам инкрементировал имя. Параметры третьего контакта придётся изменить. Во-первых, это выходной контакт, поэтому вместо имени **IN3** введём **OUT1**, а тип **Input** переправим на **Output**. Во-вторых, вывод **OUT1** не прямой, а инверсный, поэтому заменим его форму на **Dot** (вместо **Line**).

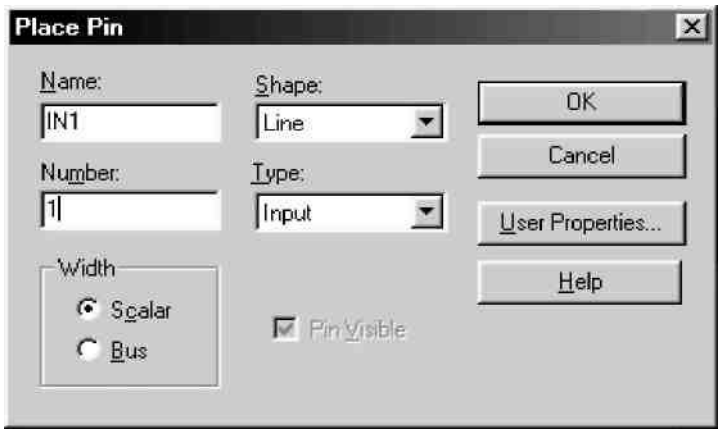


Рис. 5. *Диалоговая панель для задания параметров контакта ЭК*

Остаётся нарисовать графику символа. Активизируем команду **Place/Rectangle** и нарисуем контур символа «поверх» пунктирного прямоугольника. Командой **Place/Text** укажем функцию **&**, выполняемую электронным компонентом. Возможно, вам захочется изменить размеры текста, заданные умолчанием.

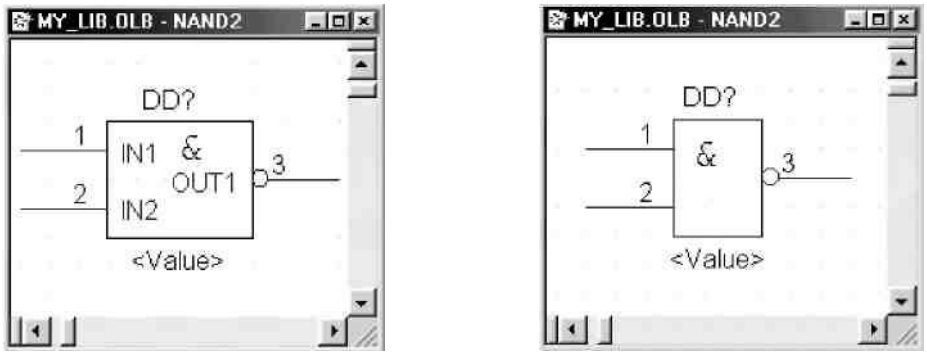
При вводе текста рекомендуется отключать режим дискретного перемещения объектов по узлам сетки, чтобы более точно позиционировать его. Для этого достаточно щёлкнуть на пиктограмме **Snap To grid**, показанной справа.

Она находится на основной панели инструментов вверху и приобретает красный цвет в режиме «плавного» перемещения курсора мыши. На самом деле в этом режиме шаг перемещения

объектов остаётся дискретным, но уменьшается до величины, равной 0.1 шага сетки. Законченный рисунок должен выглядеть так, как показано на рис. 6, *а*.

Впрочем, мы упустили одну вещь: *для простых логических элементов имена выводов очевидны*. Они не добавляют к графическому описанию новой информации, а только перегружают его ненужными деталями. Поэтому на УГО простой логики имена контактов делают невидимыми. Это даёт ещё один «плюс»: теперь на размещение имён не требуется места, поэтому символ можно перерисовать более компактно (рис. 6, *б*).

Чтобы сделать имена невидимыми, надо дважды щелкнуть левой кнопкой мыши в зоне размещения символа, не попав при этом на какой-либо объект. Появится диалоговая панель **User Properties**, на которой следует выбрать строку **Pin Names Visible** и заменить значение **True** на **False.**



а) б)

Рис. 6. *Графическое описание ЭК NAND2*

То же самое можно проделать и с номерами контактов, так как они не несут для абстрактных символов сколь либо важной информации.

Сохраним символ NAND2 и закроем окно редактора **Part and Symbol Editor**. Заметим, что двойным щелчком на созданном символе всегда можно вернуться в режим его редактирования. Если всё сделано правильно, закроем библиотеку **my_lib.olb**.

Теперь надо протестировать наше компонент и посмотреть, как оно будет выглядеть на схеме рядом с фирменными компонентами. С этой целью создадим новый проект **test_nand2**, подключим к нему библиотеку **my_lib.olb** и разместим нормальное и конвертированное изображения символа NAND2 на схеме. Рядом поместим их зарубежный аналог 7400 из библиотеки **ttl.olb** (рис. 7). Для наглядности визуализируем свойство **Graphic** всех компонентов (по умолчанию оно не выводится на экран).

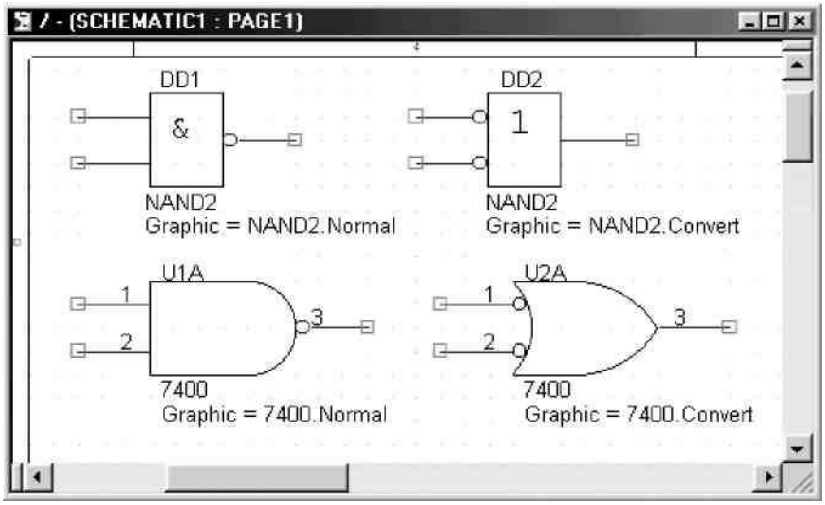


Рис. 7. *Результаты проектирования символа NAND2*

Сравнивая результат, отметим, что наши символы ничуть не хуже, они даже более компактны, а это немаловажно для больших схем.

Проектирование реальных электронных компонентов

При создании графических изображений (символов) реальных электронных компонентов требуется задать упаковочную информацию, контакты питания и земли, указать тип корпуса, в котором размещается реальный компонент.

В качестве объекта проектирования возьмём из справочника по интегральным микросхемам электронный компонент 555LA3, содержащий четыре одинаковых двухвходовых вентиля 2И-НЕ, и попытаемся «формализовать» технологию создания реального символа.

- Открываем библиотеку my_lib.olb (команда **File/Open/Library**). Вызываем диалоговую панель **New Part**

Properties (команда **Design/New Part**) и вводим следующую информацию:

- имя компонента: 555LA3;
- префикс позиционного обозначения: DD;
- типовой корпус: DIP14 (для абстрактного символа этот пункт не выполняется);
- устанавливаем флажок **Create Convert View**, чтобы создать логически эквивалентную форму графического изображения;
- устанавливаем число секций в корпусе: 4 (**Parts per Pkg**);
- оставляем тип упаковки **Homogenous**, так как в корпус помещаются одинаковые символы;
- выбираем цифровой (**Numeric**) способ нумерации секций компонента;
- щёлкаем на кнопке **OK**.

В открывшемся рабочем окне редактора символов (**Part and Symbol Editor**) создаём графический образ компонента:

- вводим входные и выходной контакты IN1, IN2, OUT1 (команда **Place/Pin**), при этом заметим, что номера контактов это не просто порядок их ввода, они должны соответствовать номерам «ножек» микросхемы, к которым подключаются входы и выход *первой* секции;
- для контакта OUT1 задаём форму вывода **Dot** (инверсный выход), причем все логические контакты делаем не видимыми (команда **Options/Part Properties**, свойство **Pin Names Visible**, значение **False**);
- вводим контакты питания **VCC** и земли **GND**, в реальной микросхеме они подаются на 14 и 7 «ножки», поэтому в поле **Number** указываем соответствующие цифры, выводы имеют нулевую длину (**Zero_Length**) и тип **Power**, флажок **Pin Visible** оставим сброшенным, так как выводы питания обычно не видны на схеме;
- задаём логическую функцию **&**, которую выполняет элемент (команда **Place/Text**) и помещаем её внутри символа;
- корректируем габаритные размеры символа («буксировкой» его углов);

- рисуем графику символа (команда **Place/Rectangle**);
- вводим упаковочную информацию для второй секции (команда **View/Next Part**), при этом придётся опять заглянуть в справочник по интегральным микросхемам, чтобы узнать, на какие «ножки» выводятся входы и выход второй секции (их номера соответственно 4, 5 и 6), а затем двойным щелчком на контакте вызываем его параметры и в пустое поле **Number** вводим нужный номер (*не забудьте проделать эту операцию и для контактов питания и земли)*;
- аналогичную работу выполняем для всех оставшихся секций;
- сохраняем созданный символ (команда **File/Save**) и закрываем библиотеку.

Протестируем полученный символ. С этой целью создадим новый проект **test_555LA3.opj** и в рабочем окне редактора схем (**Schematic Page Editor**) разместим 5 – 6 копий нашего символа (число копий должно превышать число секций в компоненте). Выполним автоматическую упаковку (команда **Tools/Annotate**… менеджера проектов). Если при вводе упаковочной информации мы не наделали ошибок, то редактору понадобится всего два корпуса, чтобы разместить все элементы. При этом номера позиционных обозначений и секций будут возрастать в направлении слева направо и сверху вниз (рис. 8).

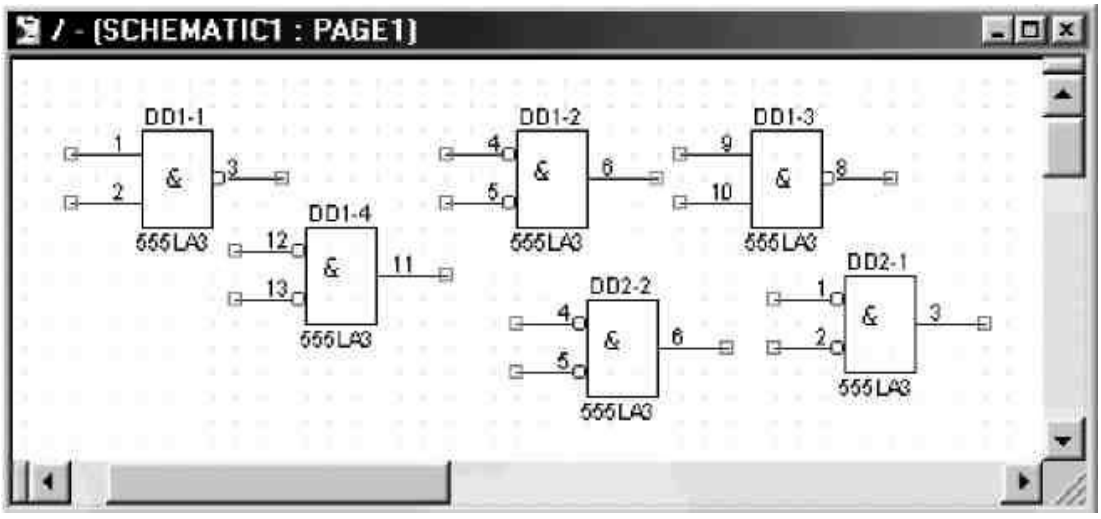


Рис. 8. *Проверка правильности задания упаковочной информации для элемента 555LA3*

Проектирование реальных компонентов с неоднородными символами

Напомним, что такие компоненты содержат в своей упаковке два или более *разных* по выполняемой функции или по числу внешних выводов элементов. Например, компонент 555LR11 содержит два элемента И-ИЛИ-НЕ с разным числом входов, а в микросхеме 564LP2 объединены два элемента 3ИЛИ-НЕ и один инвертор НЕ. В таких случаях говорят, что компонент содержит секции разного типа (**Heterogeneous**).

Структурная схема алгоритма создания символа электронного компонента

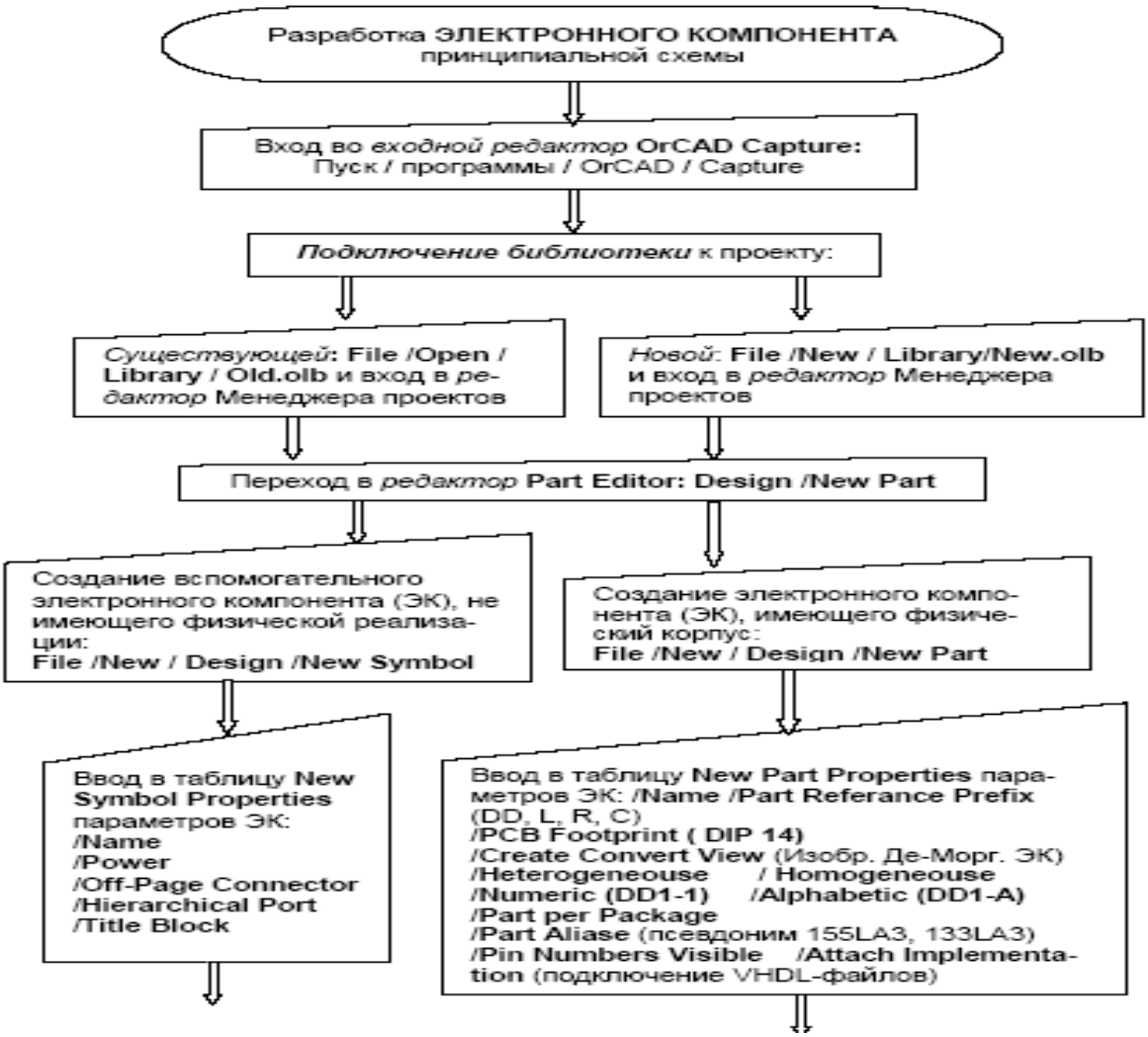


Рис. 9. *Структурная схема алгоритма создания символа электронного компонента принципиальной схемы* ***с*** *помощью графического редактора* ***Part Edit***

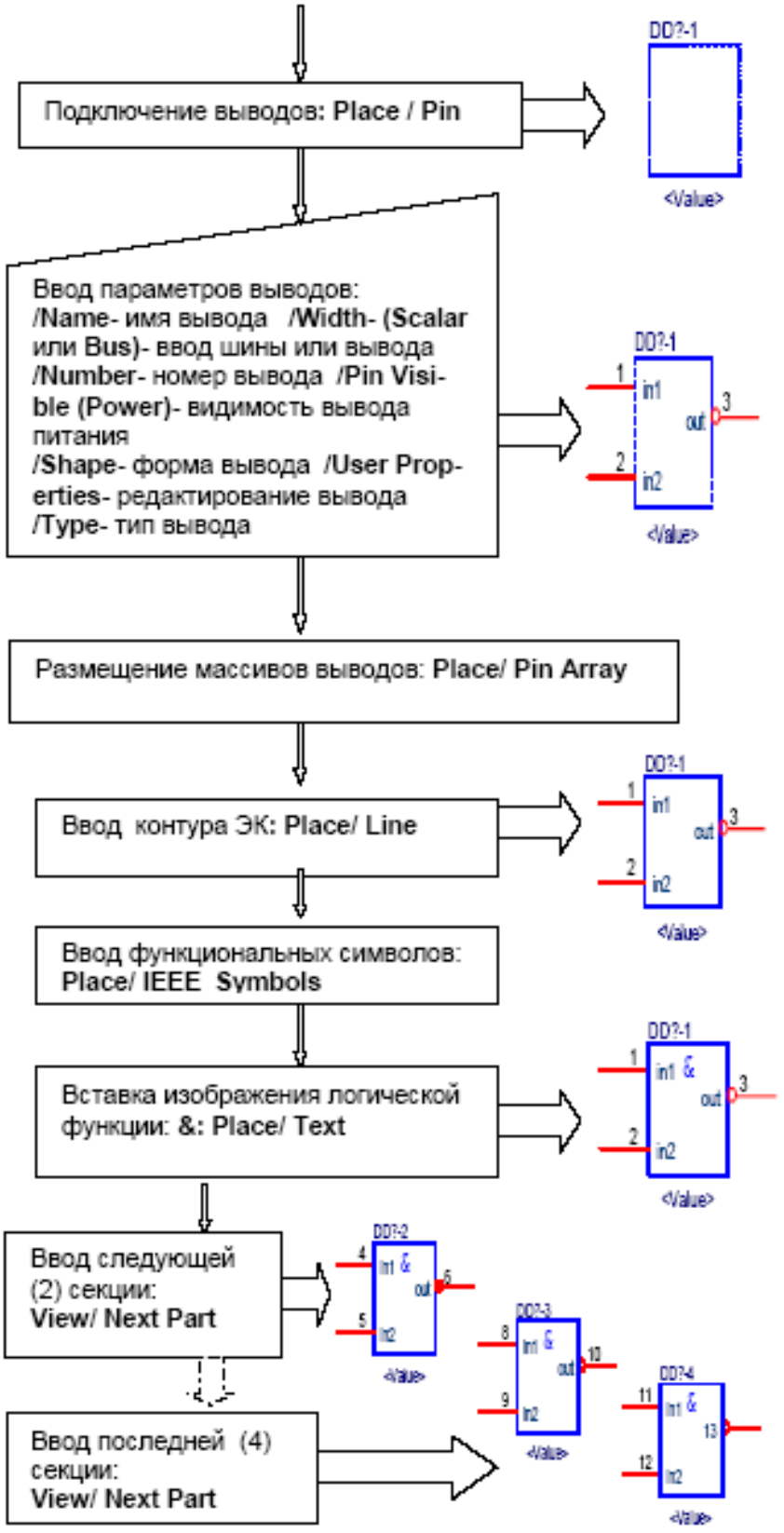


Рис. 9. Продолжение

В заключение на рис. 9 приведена ***структурная схема*** алгоритма ***создания символа электронного компонента*** принципиальной схемы с помощью графического редактора **Part Editor**.

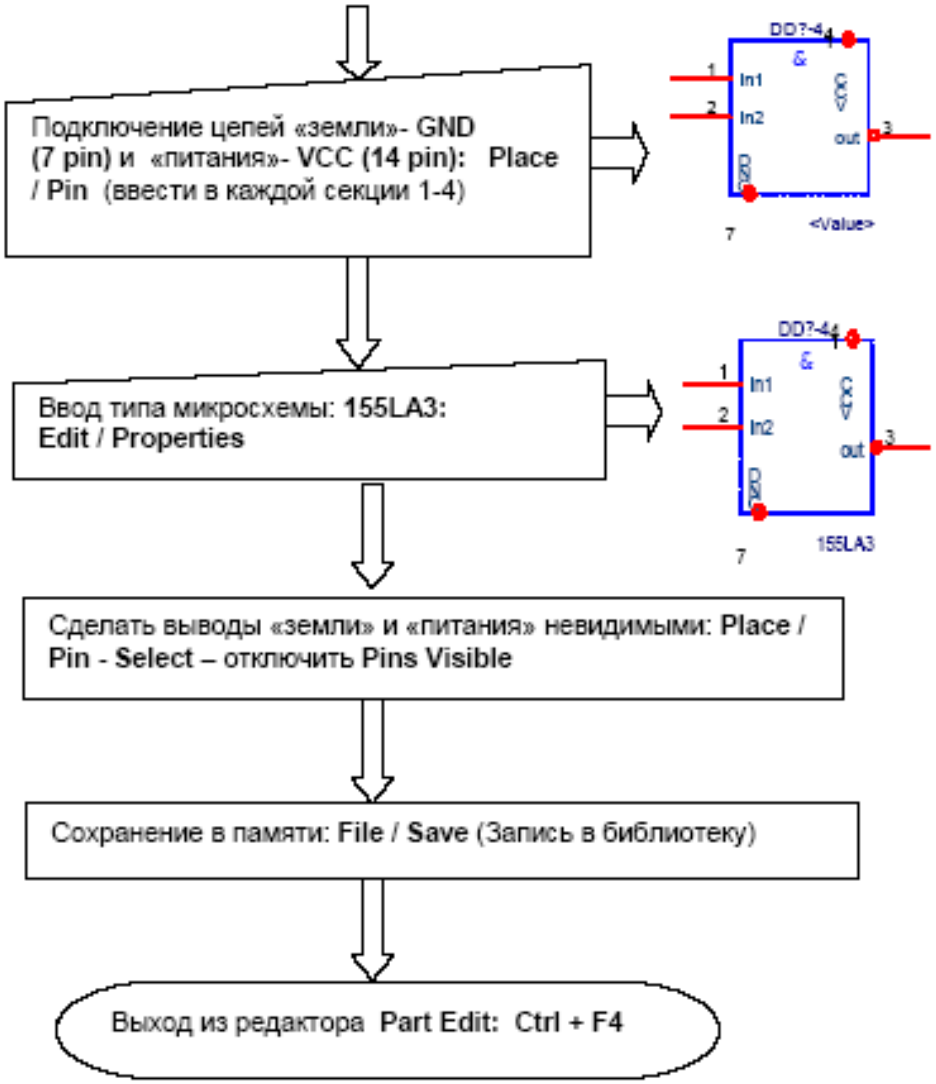


Рис. 9. Окончание

Как следует из структурной схемы, основными этапами проектирования электронного компонента принципиальной схемы являются:

– загрузка входного редактора OrCAD Capture по команде Пуск/Программы/OrCAD/Capture;
– подключение к проекту библиотеки по командам

File/Open/Library/Old.olb (существующая библиотека) или **File/New/Library/New.olb** (новая библиотека) и автоматический вход в *редактор* Менеджера проектов;

- переход из **Менеджера проектов** в *редактор* **Part Editor** для создания электронных компонентов с физическим корпусом по-команде ГМ **Design/New Part** или по команде ГМ **Design /New Symbol** для создания вспомогательного электронного компонента, не имеющего физической реализации;
- ввод в таблицы **New Part Properties** или **New Symbol Properties** параметров физических или вспомогательных ЭК;
- подключение к электронному компоненту выводов по команде ГМ **Place/Pin** и ввод параметров выводов в таблицу по команде ГМ **Place Pin**;
- размещение массивов выводов по команде ГМ **Place/Pin Array**;
- ввод контура ЭК: **Place/Line**;
- ввод функциональных символов ЭК по команде ГМ **Place/IEEE Symbols**;
- вставка изображения логической функции: **&** по команде ГМ **Place/Text**;
- ввод секций ЭК по команде ГМ **View/Next Part**;
- подключение цепей «земли» – **GND (7 pin)** и «питания» – **VCC (14 pin)** по команде ГМ **Place/Pin** (ввод в каждую секцию 1 – 4);
- ввод типа микросхемы **155LA3** по команде ГМ **Edit/Properties**;
- сделать выводы «земли» и «питания» невидимыми – с помощью команды ГМ **Place/Pin** в таблице **Place Pin** отключить **Pins Visible**;
- сохранение в памяти: **File/Save** (запись в библиотеку) и выход из редактора **Part Edit** по команде клавиатуры **Ctrl + F4**.

3. ПОДГОТОВКА К РАБОТЕ

Изучить методические указания к данной работе.

Установить пакет OrCAD в собственном персональном компьютере или освоить работу с пакетом в лаборатории кафедры.

Выполнить приведенные в тексте методических указаний примеры.

4. РАБОЧЕЕ ЗАДАНИЕ

В справочнике по радиоэлектронным компонентам выбрать по указанию преподавателя пассивные (R, C, L) и активные (интегральные микросхемы) электронные компоненты для создания их библиотечных символов в пакете OrCAD.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные параметры, свойства и структура пакета OrCAD.
2. Перечислите функции менеджера проекта.
3. Изобразите структуру и опишите назначение команд главного меню графического редактора Part Editor.
4. Опишите методику и особенности проектирования символов электронных компонентов.
5. Как создать символы электронных компонентов?
6. Как создать и спроектировать абстрактные электронные компоненты?
7. Расскажите об особенностях создания VHDL-модели электронного компонента.
8. Опишите алгоритм и изобразите структурную схему создания реальных символов электронных компонентов.
9. Перечислите основные этапы проектирования реальных электронных компонентов.
10. Перечислите основные этапы проектирования реальных компонентов с неоднородными символами.
11. Изобразите структурную схему алгоритма создания символа электронного компонента.
12. Как отредактировать символы электронных компонентов?
13. Как вывести на печать изображение символов электронных компонентов?