MAX+PLUS II Общие сведения

До последнего времени **MAX+PLUS II** являлся единственной системой проектирования устройств на ПЛИС **Altera**. Только в 1999 году появилась система проектирования нового поколения **Quartus**, предназначенной для проектирования устройств на ПЛИС APEX20K. ПО системы MAX+PLUS II, представляющее собой единое целое, обеспечивает управление пользователя средой логического проектирования и помогает достичь максимальной эффективности и производительности. Все пакеты работают как на платформе IBM PC, так и на платформах SUN, IBM RISC/6000 и HP9000. В дальнейшем мы будем рассматривать работу на платформе IBM PC.

Для нормальной инсталляции и работы САПР **MAX+PLUS II** необходима IBM PC совместимая ЭВМ с процессором не хуже Pentium, объемом ОЗУ не хуже 16 Мб и свободным местом на жестком диске порядка 150 –400 Мб в зависимости от конфигурации системы. Из собственного опыта можем сказать, что для разработки больших кристаллов на ПЛИС FLEX10K50 и выше желательно иметь не менее 64 Мб ОЗУ (лучше 128 еще лучше 256, совсем хорошо 384 Мб и выше) и процессор Pentium II (P-3 реально не дает особого выигрыша). Конечно, можно использовать и более слабые машины, но тогда возрастает время компиляции и увеличивается нагрузка на жесткий диск из-за свопинга. Увеличение объема оперативной памяти и кэша дает лучшие результаты по сравнению с увеличением тактовой частоты процессора. Если не предполагается трассировка больших кристаллов, то вполне хватает 32 Мб ОЗУ при хорошей скорости компиляции проекта. Что касается выбора операционной системы, то без сомнения, лучше использовать Windows NT, хуже Windows 95 OSR2, плохо Windows 98, особенно локализованную версию. Очевидно, это связано с тем, что изначально пакет был разработан под Unix и не полностью использует все механизмы Windows. Особенно это заметно при временном моделировании сложных устройств ЦОС, когда перерисовка экрана занимает основное время. Поскольку пакет не локализован, то лучше использовать не локализованные (американскую или паневропейскую) версии Windows.

Во время инсталляции системы **MAX+PLUS II** создаются два каталога: **\maxplus2** и **\max2work.** Каталог **\maxplus2** содержит системное ПО и файлы данных и разбит на подкаталоги, перечисленные в табл.2.1.

Таблица 2.1. Структура системного каталога \maxplus2 системы MAX+PLUS II

Подкаталог	Описание
.\drivers	Содержит драйверы устройств для среды WINDOWS NT (только для инсталляции на платформе PC в среде WINDOWS NT)
.\edc	Содержит поставляемые фирмой Altera командные файлы (.edc), которые генерируют выходные файлы (.edo) по заказу пользователя для заданных условий тестирования
.\lmf	Содержит поставляемые фирмой Altera файлы макробиблиотек (.lmf), которые устанавливают соответствие между логическими функциями пользователя и эквивалентными логическими функциями MAX+PLUS II
.\max2inc	Содержит Include-файлы (файлы “заголовков”) с прототипами функций для разхработанных фирмой Altera макрофункций. В прототипах функций перечисляются порты (выводы) для макрофункций, реализованных в текстовых файлах проекта (.tdf), написанных на языке AHDL
.\max2lib\edif	Содержит примитивы и макрофункции, используемые для пользовательских интерфейсов EDIF
.\max2lib\mega_lpm	Содержит мегафункции, в том числе библиотеку функций параметризованных модулей (LPM) и Include-файлы для них с соответствующими прототипами на языке AHDL
.\max2lib\mf	Содержит макрофункции пользовательские и устаревшие (74-series)

.\max2lib\prim	Содержит поставляемые фирмой Altera примитивы
.\vhdlnn\altera	Содержит библиотеку altera с программным пакетом maxplus2. В этот пакет входят все примитивы, мегафункции и макрофункции системы MAX+PLUS II, поддерживаемые языком VHDL
.\vhdlnn\ieee	Содержит библиотеку ieee пакетов VHDL, в том числе std_logic_1164, std_logic_arith, std_logic_signed и std_logic_unsigned
.\vhdlnn\std	Содержит библиотеку std с пакетами стандартов и средств ввода/вывода текста, описанными в справочнике по стандартам института IEEE на языке VHDL IEEE Standard VHDL Language Reference Manual

Каталог **\max2work** содержит файлы обучающей программы и примеры и разделяется на подкаталоги, описанные в табл. 2.2.

Таблица 2.2. Структура рабочего каталога \max2work системы MAX+PLUS II

Подкаталог	Описание
.\ahdl	Содержит файллы примеров, иллюстрирующих тему “Как использовать язык AHDL” (How to Use AHDL) в электронном справочнике (MAX+PLUS II Help) и в руководстве MAX+PLUS II AHDL
.\chiptrip	Содержит все файлы обучающего проекта chiptrip, описанного в руководстве MAX+PLUS II AHDL
.\edif	Содержит все файлы примеров, иллюстрирующих особенности EDIF в электронном справочнике (MAX+PLUS II Help)
.\tutorial	Содержит информационный файл read.me обучающего проекта chiptrip. Все файлы, сохдаваемые в проекте chiptrip, должны находиться в этом подкаталоге
.\vhdl	Содержит файлы примеров, иллюстрирующих тему “Как использовать язык VHDL” (How to Use VHDL) в электронном справочнике (MAX+PLUS II Help) и в руководстве MAX+PLUS II VHDL
.\verilog	Содержит файлы примеров, иллюстрирующих тему “Как использовать язык верификационного протокола Verilog HDL” (How to Use Verilog HDL) в электронном справочнике (MAX+PLUS II Help) и в руководстве MAX+PLUS II Verilog HDL

Название системы MAX+PLUS II является аббревиатурой от **M**ultiple **A**rray Matri**X** **P**rogrammable **L**ogic **U**ser **S**ystem (Пользовательская система программирования логики упорядоченных структур). Система MAX+PLUS II разработана фирмой Altera и обеспечивает многоплатформенную архитектурно независимую среду создания дизайна, легко приспосабливаемую для конкретных требований пользователя. Система MAX+PLUS II имеет средства удобного ввода дизайна, быстрого прогона и непосредственного программирования устройств.

Представленный на рис. 2.1 состав ПО системы MAX+PLUS II является полным комплектом, обеспечивающим создание логических дизайнов для устройств фирмы Altera с программируемой логикой, в том числе семейства семейства устройств Classic, MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000 и FLEX 10K. Информация о других, поддерживаемых семействах устройств фирмы Altera приведена в файле **read.me** в системе MAX+PLUS II.

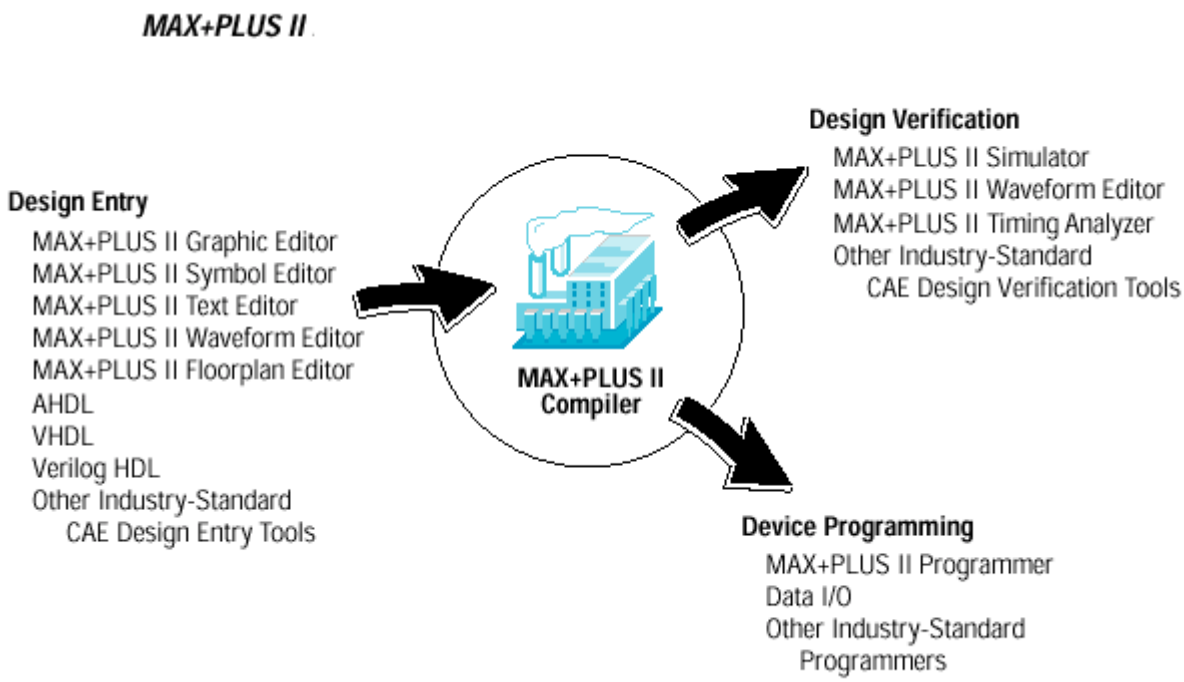

Рис. 2.1. Среда проектирования в системе MAX+PLUS II

Система MAX+PLUS II предлагает полный спектр возможностей логического дизайна: разнообразные средства описания проекта для создания проектов с иерархической структурой, мощный логический синтез, компиляцию с заданными временными параметрами, разделение на части, функциональное и временное тестирование (симуляцию), тестирование нескольких связанных устройств, анализ временных параметров системы, автоматическую локализацию ошибок, а также программирование и верификацию устройств. В системе MAX+PLUS II можно как читать, так и записывать файлы на языке AHDL и файлы трассировки в формате EDIF, файлы на языках описания аппаратуры Verilog HDL и VHDL а также схемные файлы OrCAD. Кроме того, система MAX+PLUS II читает файлов трассировки, созданных с помощью ПО Xilinx, и записывает файлы задержек в формате SDF для удобства взаимодействия с пакетами, работающими с другими промышленными стандартами.

Система MAX+PLUS II предлагает пользователю богатый графический интерфейс, дополненный иллюстрированной оперативной справочной системой. В полную систему MAX+PLUS II входят 11 полностью внедренных в систему приложений (рис. 2.2). (Логический дизайн (design), включая все поддизайны (subdesign), называется в системе MAX+PLUS II проектом (project))

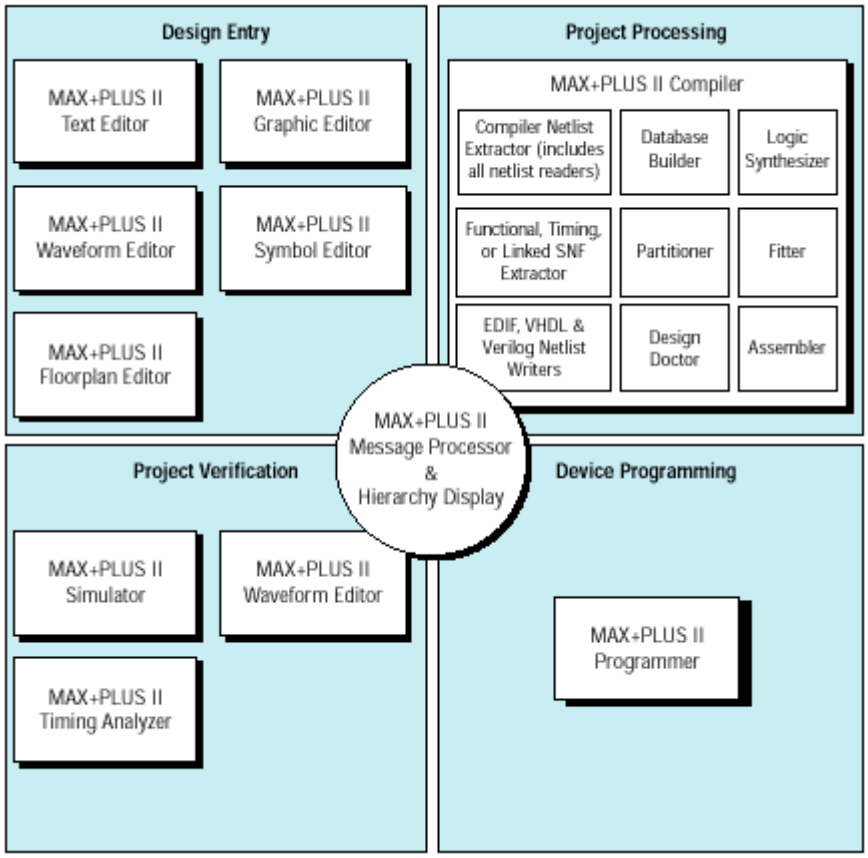

Рис. 2.2. Приложения в системе MAX+PLUS II

Для ввода описания проекта (Design Entry) возможно описание проекта в виде файла на языке описания аппаратуры, созданного либо во внешнем редакторе, либо в текстовом редакторе MAX+PLUS II (Text Editor), в виде схемы электрической принципиальной с помощью графического редактора Graphic Editor, в виде временной диаграммы, созданной в сигнальном редакторе Waveform Editor. Для удобства работы со сложными иерархическими проектами каждому поддизайну может быть сопоставлен символ, редактирование которого производится с помощью графического редактора Symbol Editor. Размещение узлов по ЛБ и выводам ПЛИС выполняют с помощью поуровневого планировщика Floorplan Editor.

Верификация проекта (Project verification) выполняется с помощью симулятора (simulator), результаты работы которого удобно просмотреть в сигнальном редакторе Waveform Editor, в нем же создаются тестовые воздействия.

Компиляция проекта, включая извличение списка соединений (Netlist Extractor), построение базы данных проекта (Data Base Builder), логический синтез (logic synthesis), извлечение временных, функциональных параметров проекта (SNF Extractor), разбиение на части (Partioner), трассировка (Fitter) и формирование файла программирования или загрузки (Assembler) выполняются с помощью компилятора системы (Compiler)/

Непосредственно программирование или загрузка конфигурации устройств с использованием соответствующего аппаратного обеспечения выполняется с использованием модуля программатора (Programmer).

Многие характерные черты и команды – такие как открытие файлов, ввод назначений устройств, выводов и логических элементов, компиляция текущего проекта – похожи для многих приложений системы MAX+PLUS II. Редакторы для разработки проекта (графический, текстовый и сигнальный) имеют много общего со вспомогательными редакторами (поуровневого планирования и символьный). Каждый редактор разработки проекта позволяет выполнять похожие задачи (например, поиск сигнала или символа) похожим способом. Можно легко комбинировать разные типы файлов проекта в иерархическом проекте, выбирая для каждого функционального блока тот формат описания проекта, который больше подходит. Поставляемая фирмой Altera большая библиотека мега- и макрофункций, в том числе функции из библиотеки параметризованных моделей (LPM), обеспечивает широкие возможности ввода дизайна

Можно одновременно работать с разными приложениями системы MAX+PLUS II. Например, можно открыть несколько файлов проекта и переносить информацию из одного в другой в процессе компиляции или тестирования другого проекта. Или например, просматривать все дерево проекта и в окне просмотра перемещаться с одного уровня на другой, а в окне

редактора будет появляться выбранный вами файл, причем вызывается автоматически соответствующий редактор для каждого файла (рис.2.3)

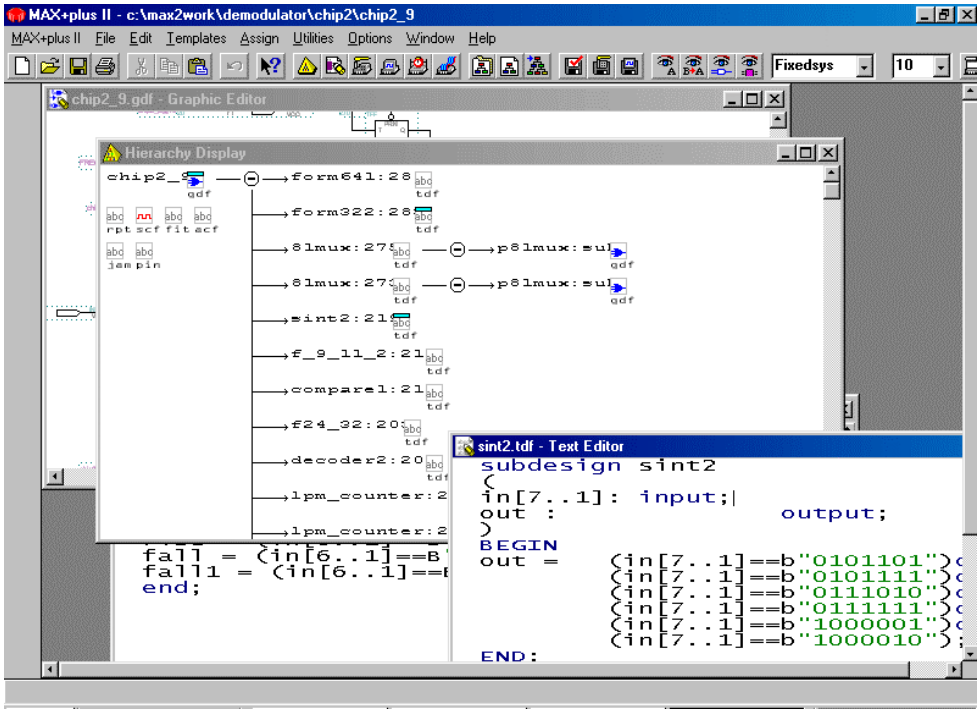

Рис.2.3. Иерархический просмотр проекта

Основой системы MAX+PLUS II является компилятор, обеспечивающий мощные средства обработки проекта, при этом можно задавать нужные режимы работы компилятора. Автоматическая локализация ошибки, выдача сообщения и обширная документация об ошибках ускоряют и облегчают проведение изменений в дизайне. Можно создавать выходные файлы в разных форматах для разных целей, таких как работа функций, временных параметров и связи нескольких устройств; анализа временных параметров; программирования устройства.

2.2 Процедура разработки проекта

Процедуру разработки нового проекта (project) от концепции до завершения можно упрощенно представить следующим образом:

- создание нового файла (design file) проекта или иерархической структуры нескольких файлов проекта с использованием различных редакторов разработки проекта в системе MAX+PLUS II, т.е. графического, текстового и сигнального редакторов;

- задание имени файла проекта верхнего уровня (Top of hierarchy) в качестве имени проекта (Project name);

- назначение семейства ПЛИС для реализации проекта. Пользователь может сам назначить конкретное устройство или предоставить это компилятору для того, чтобы оценить требуемые ресурсы;

- открытие окна компилятора (**Compiler)** и его запуск нажатием кнопки **Start** для начала компиляции проекта. По желанию пользователя можно подключить модуль извлечения временных задержек Timing SNF Extractor для создания файла разводки, используемого при тестировании временных параметров и анализе временных параметров;

- в случае успешной компиляции возможно тестирование и временной анализ, для проведения которого необходимо выполнить следующие действия:

- для проведения временного анализа открыть окно **Timing Analyzer**, выбрать режим анализа и нажать кнопку **Start**;
- для проведения тестирования нужно сначала создать векторной тестовый вектор в файле канала тестирования (**.scf**), пользуясь сигнальным редактором, или в файле вектора (.vec), пользуясь текстовым редактором. Затем открыть окно отладчика – симулятора (**Simulator)** и нажать кнопку **Start**;

- программирование или загрузка конфигурации выполняется путем запуска модуля программатора (**Programmer)** с последующей вставкой устройства в программирующий адаптер программатора MPU (Master Programming Unit) или подключение устройств MasterBlaster, BitBlaster, ByteBlaster или кабеля загрузки FLEX (FLEX Download Cable) к устройству, программируемому в системе;
- выбор кнопки **Program** для программирования устройств с памятью типа EPROM или EEPROM (MAX, EPC) либо выбор кнопки **Configure** для загрузки конфигурации устройства с памятью типа SRAM (FLEX).

Ниже будут подробно рассмотрены основные элементы разработки проекта в системе MAX+PLUS II.

Систему MAX+PLUS II можно запустить двумя способами щелкнув дважды левой кнопкой мыши на пиктограмме MAX+PLUS II) или набрав **maxplus2** в командной строке.

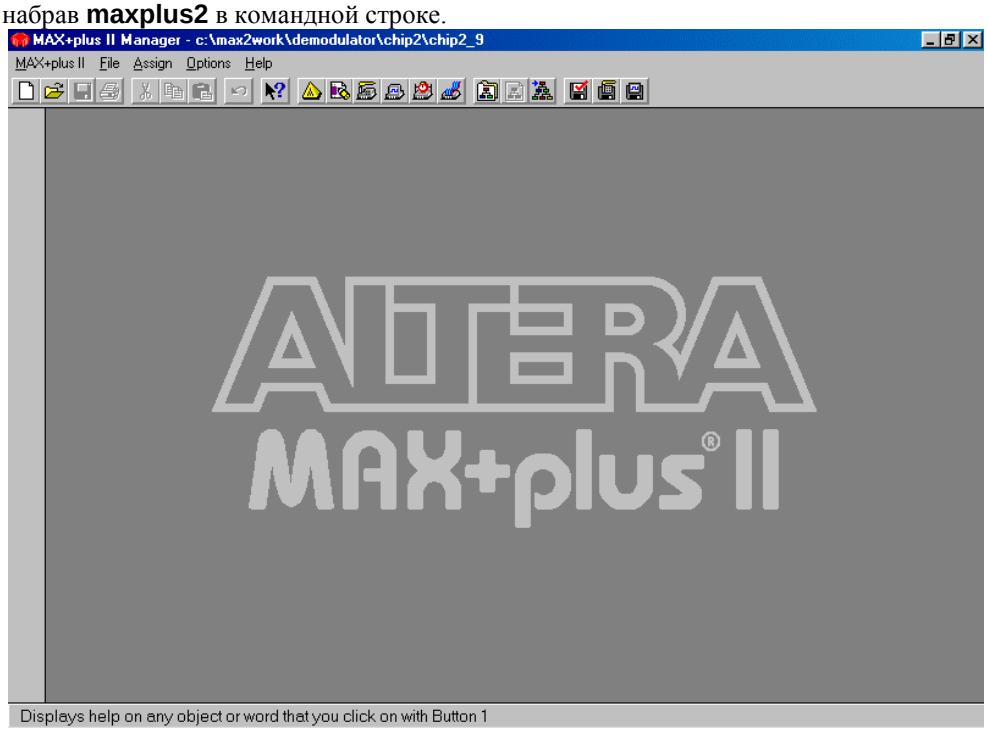

Рис. 2.4. Главное окно системы MAX+PLUS II

При запуске системы MAX+PLUS II автоматически открывается ее Главное окно, меню которого охватывает все приложения системы MAX+PLUS II (см. рис. 2..4).

В верхней части окна отображается имя проекта и текущего файла проекта. Затем следует строка меню, под ней панель основных инструментов системы, обеспечивающая быстрый вызов ее компонентов.

В нижней части экрана располагается строка подсказки.

Вызов компонентов системы удобно производить MAX+PLUS II, представленном на рис.2.5.

Рис. 2.5. Окно меню MAX+PLUS II

Рассмотрим подробнее меню MAX+PLUS II (рис. 2.5). ПО системы MAX+PLUS II содержит 11 приложений и главную управляющую оболочку. Различные приложения, обеспечивающие создание файлов проекта, могут быть активизированы мгновенно, что позволяет пользователю переключаться между ними щелчком мыши или с помощью команд меню. В это же время может работать одно из фоновых приложений, например компилятор, симулятор, временной анализатор и программатор. Одни и те же команды разных приложений работают одинаково, что облегчает задачу разработки проекта.

Окно любого приложения можно свернуть до пиктограммы, не закрывая приложения, а затем снова развернуть его. Это позволяет пользователю работать эффективно, не загромождая рабочий экран.

В табл. 2.3 приведены пиктограммы и описание приложений.

Таблица 2.3

Приложения системы MAX+PLUS II

Приложение	Выполняемая функция
Hierarchy Display 	**Обзор иерархии –** отображает текущую иерархическую структуру файлов в виде дерева с ветвями, представляющими собой поддизайны. Можно визуально определить, является ли файл проекта схемным, текстовым или сигнальным; какие файлы открыты в данный момент; какие вспомогательные файлы в проекте доступны пользователю для редактирования. Можно также непосредственно открыть или закрыть один или несколько файлов дерева и ввести назначения ресурсов для них
Graphic Editor 	**Графический редактор –** позволяет разрабатывать схемный логический дизайн в формате реального отображения на экране WYSIWYG. Применяя разработанные фирмой Altera примитивы, мегафункции и макрофункции в качестве основных блоков разработки, пользователь может также использовать собственные символы
Symbol Editor 	**Символьный редактор –** позволяет редактировать существующие символы и создавать новые
Text Editor 	**Текстовый редактор –** позволяет создавать и редактировать текстовые файлы проекта, написанные на языках описания аппаратуры AHDL, VHDL и Verilog HDL. Кроме того, в этом редакторе можно создавать, просматривать и редактировать другие файлы формата ASCII, используемые другими приложениями MAX+PLUS II. Можно создавать файлы на языках HDL и в других текстовых редакторах, однако данный текстовый редактор системы MAX+PLUS II дает преимущества в виде контекстной справки, выделения цветом синтаксических конструкций и готовых шаблонов языков AHDL, VHDL и Verilog HDL
Waveform Editor 	**Сигнальный редактор –** выполняет двойную функцию: инструмент для разработки дизайна и инструмент для ввода тестовых сигналов и наблюдения результатов тестирования
Floorplan Editor 	**Поуровневый планировщик –** позволяет графическими средствами делать назначения выводам устройства и ресурсов логических элементов и блоков. Можно редактировать расположение выводов на чертеже корпуса устройства и назначать сигналы отдельным логическим элементам на более подробной схеме логической структуры (LAB view). Можно также просматривать результаты последней компиляции
Compiler	**Компилятор –** обрабатывает логические проекты, разработанные для семейств устройств Altera Classic. MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000 и FLEX 10K. Большинство заданий выполняется автоматически. Однако пользователь может управлять процессом компиляции

	полнеостью или частично
Simulator 	**Симулятор –** позволяет тестировать логические операции и внутреннюю синхронизацию проектируемой логической схемы. Возможны три режима тестирования: функциональное, временное и тестирование нескольких соединенных между собой устройств
Timing Analyzer 	**Анализатор временных параметров –** анализирует работу проектируемой логической цепи после того, как она была синтезирована и оптимизирована компилятором, позволяет оценить задержки, возникающие в схеме.
Programmer 	**Программатор –** позволяет программировать, конфигурировать, проводить верификацию и испытывать устройства фирмы Altera
Message Processor 	**Генератор сообщений –** выдает на экран сообщения об ошибках, предупреждающие и информационные сообщения о состоянии проекта пользователя и позволяет пользователю автоматически найти источник сообщения в исходном или вспомогательном файле (файлах) и в поуровневом плане назначений

На рис. 2.3 проиллюстрирован рабочий момент многооконной разработки дизайна, когда на экране одновременно открыты окно обзора иерархии и текстового редактора.

Перед тем как начать работать в системе MAX+PLUS II, следует понять разницу между файлами проекта (design-file), вспомогательными файлами и проектами.

Файл проекта– это графический, текстовый или сигнальный файл, созданный с помощью графического или сигнального редакторов системы MAX+PLUS II или в любом другом использующем промышленные стандарты схемном или текстовом редакторе либо при помощи программы **netlist writer**, имеющейся в пакетах, поддерживающих EDIF, VHDL и Verilog HDL. Этот файл содержит логику для проекта MAX+PLUS II и обрабатывается компилятором. Компилятор может автоматически обрабатывать следующие файлы проекта:

графичесие файлы проекта (**.gdf**);

текстовые файлы проекта на языке AHDL (**.tdf**);

сигнальные файлы проекта (**.wdf**);

файлы проекта на языке VHDL (**.vhd**);

файлы проекта на языке Verilog (**.v**);

схемные файлы OrCAD (**.sch**);

входные файлы EDIF (**edf**);

файлы формата Xilinx Netlist (**.xnf**);

файлы проекта Altera (**.adf**);

файлы цифрового автомата (**.smf**).

Вспомогательные файлы – это файлы, связанные с проектом MAX+PLUS II, но не являющиеся частью иерархического дерева проекта. Большинство таких файлов не содержит логики дизайна. Некоторые из них создаются автоматически приложением системы MAX+PLUS II, другие – пользователем. Примерами вспомогательных файлов являются файлы назначений и конфигурации (**.acf**), символьные файлы (**.sym**), файлы отчета (**.rpt**) и файлы тестовых векторов (**.vec**).

Проект состоит из всех файлов иерархической структуры дизайна, в том числе вспомогательных и выходных файлов. Именем проекта является имя файла ghjtrnf верхнего уровня без расширения. Система MAX+PLUS II выполняет компиляцию, тестирование, временной анализ и программирование сразу целого проекта, хотя пользователь может в это время редактировать файлы этого проекта в рамках другого проекта. Например, во время компиляции проекта **project1** пользователь может редактировать дизайн файл его TDF, который является также файлом проекта **project2** и сохранять его; однако если он захочет компилировать его, нужно будет сначала задать имя **project2** в качестве имени проекта.

Для каждого проекта следует создавать отдельный подкаталог в рабочем каталоге системы MAX+PLUS II (**\max2work**).

В системе MAX+PLUS II легко доступны все инструменты для создания логического проекта. Разработка проекта ускоряется за счет имеющихся стандартных логических функций, в том числе примитивов, мегафункций, библиотеки параметризированных модулей LPM и макрофункций устаревшего типа микросхем 74 серии. Крайне вредно использовать устаревшие библиотеки и typ=no переносить на ПЛИС схемотехнику стандартных ТТЛ серий. Следует проектировать проект именно под архитектуру ПЛИС для получения более менее разумных результатов.

Схемные файлы проекта создаются в графическом редакторе MAX+PLUS II. Можно также открыть, редактировать и сохранять схемы, созданные схемным редактором OrCAD.

Проекты на языках AHDL, VHDL и Verilog HDL создаются в текстовом редакторе MAX+PLUS II или любом другом текстовом редакторе.

Сигнальные проекты создаются в сигнальном редакторе MAX+PLUS II.

Файлы форматов EDIF и Xilinx, разработанные другими стандартными инструментами системы EDA, могут быть импортированы в среду MAX+PLUS II.

Схемные и тестовые файлы, созданные в системе MAX+PLUS II (под ДОС) и программных пакетах фирмы Altera A+PLUS и SAM+PLUS могут быть интегрированы в среде MAX+PLUS II.

Назначения физических ресурсов для любого узла или контакта в текущем проекте могут быть введены в графическую среду с помощью поуровневого планировщика. Он сохраняет для проекта назначения в файле с расширением **.acf**, в котором хранятся все типы назначений ресурсов, зондов (Probes) и устройств (Devices) так же, как и конфигурационные установки (Assign) для компилятора, симулятора и временного анализатора.

Графические символы, представляющие любой тип файла проекта могут быть автоматически созданы в любом из редакторов MAX+PLUS II, предназначенных для разработки проектов в помощью команды File/Create Default Symbol Command. С помощью символьного редактора MAX+PLUS II можно редактировать символы или создавать собственные, а затем использовать их в любом схемном файле проекта.

В иерархической структуре проекта на любом уровне допускается смешанное использование файлов с расширениями **.gdf .tdf .vhd .v .edf .sch** Однако файлы с расширением **.wdf .xnf .adf .smf** должны быть либо на самом нижнем иерархическом уровне проекта, либо быть единственным файлом.

Способы задания файлов проекта показаны на рис.2.6.

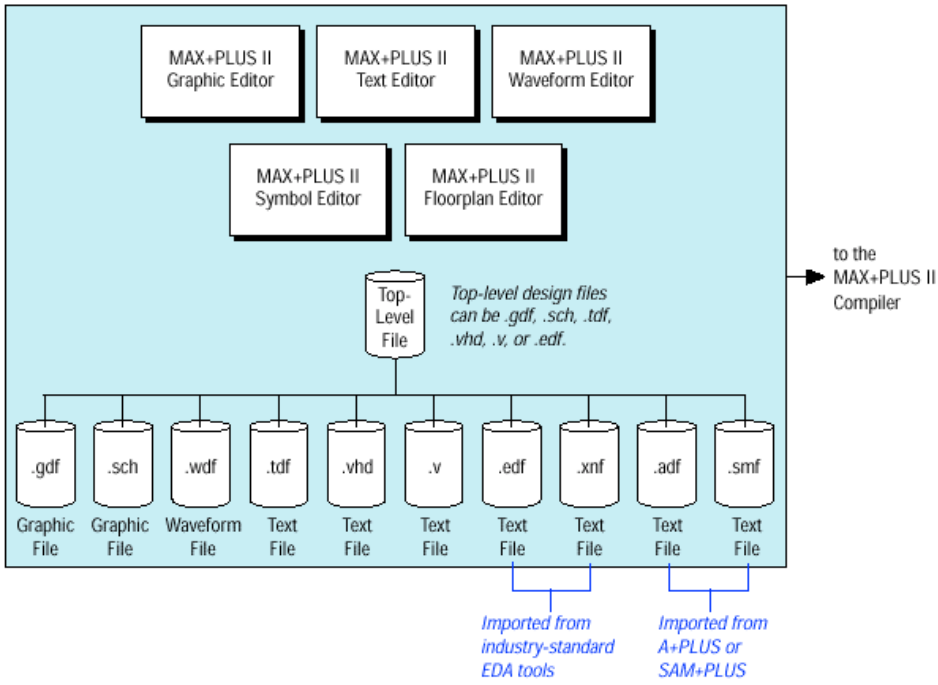

Рис. 2.6. Способы описания файлов проекта

Во всех приложениях MAX+PLUS II есть возможность с помощью команд из меню **Assign** (Назначить) вводить, редактировать и удалять типы назначений ресурсов, устройств и параметров, которые управляют компиляцией проекта, в том числе логическим синтезом, разделением на части и подгонкой. На рис. 2.7 представлены команды меню **Assign**. Пользователь может делать назначения для текущего проекта независимо от того, открыт ли какой-нибудь файл проекта или окно приложений. Система MAX+PLUS II сохраняет информацию для проекта в файле с расширением **.acf**. Изменения назначений, сделанные в окне поуровневого планировщика, также сохраняются в файле ACF. Кроме того, можно редактировать файл ACF для проекта в текстовом редакторе.

Рис. 2.7. Меню назначений проекта Assign.

Следующие функции являются общими для всех приложений MAX+PLUS II: назначения устройств, ресурсов и зондов; сохранение предыдущей версии; глобальные опции устройства в проекте; глобальные параметры проекта. глобальные

требования по временных параметров проекта; глобальный логический синтех проекта. Рассмотрим их подробнее.

Ресурс является частью устройства фирмы Altera, как например контакт или логический элемент, который выполняет конкретное, определенное пользователем задание. Пользователь может назначить логику ресурсам устройства для гарантии того, что компилятор MAX+PLUS II сделает подгонку в проекте так, как хочет пользователь. Есть следующие типы назначений.

Clique assignment (Назначение клики) задает, какие логические функции должны оставаться вместе. Группировка логических функций в клики гарантирует, что они реализуются в одном и том же блоке логической структуры LAB, блоке ячеек памяти EAB, в одном ряду или устройстве. Для назначения клики используют команду Assign/ Clique (рис.2.8)

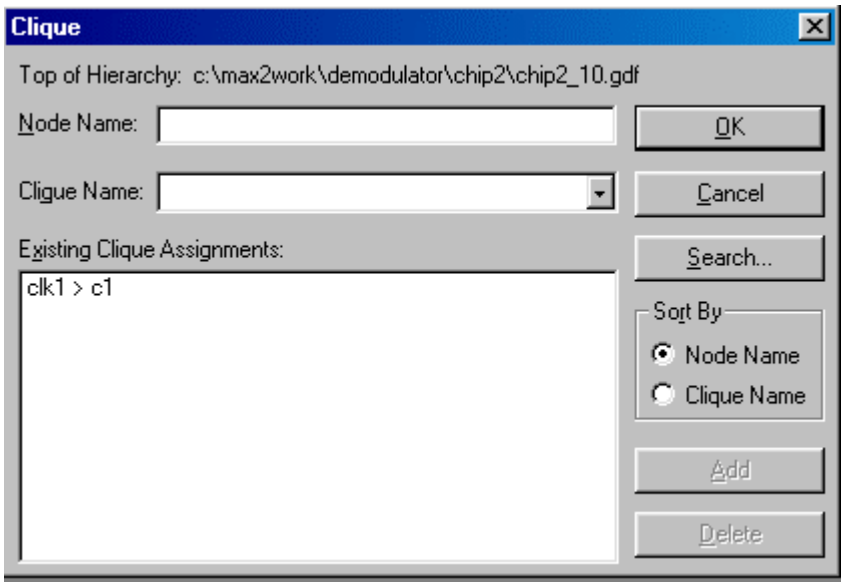

Рис.2.8. Назначение клики(команда Assign/ Clique)

В поле clique name задают имя клики. В поле Node name задают имя узла. Узлы, объединенные в клики (иногда называют группы, но такое наименование приводит к путанницей с понятием группы в AHDL), при компиляции будут размещаться в пределах одного ЛБ

Chip assignment (Назначение чипа) задает, какие логические функции должны быть реализованы в одном и том же устройстве в случае разделения проекта на части (на нескольео устройств).

Pin assignment (Назначение вывода) назначает вход или выход одной логической функции, такой как примитив или мегафункция, конкретному контакту или вертикальному (горизонтальному) ряду выводов ПЛИС.

Location assignment (Назначение ячейки) назначает единственную логическую функцию, такую как выход примитива или мегафункции, конкретной ячейке чипа, такой как логический элемент, ячейка ввода/вывода, ячейка памяти, блоки LAB и EAB, горизонтальные или вертикальные ряды.

Назначения вывода, чипа и ячейки выполняется с помощью команды Assign/ Pin/Location/Chip, окно которой приведено на рис.2.9

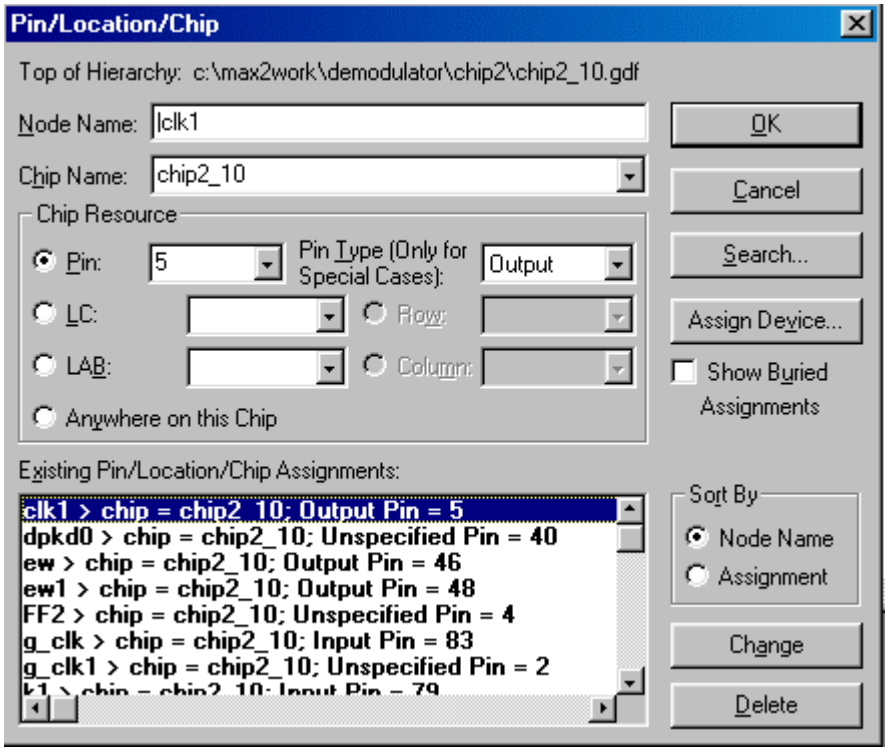

Рис.2.9. Окно команды Assign/ Pin/Location/Chip

В полях данного окна можно задать номер вывода (Pin), логическую ячейку (LC) или блок (LAB), а также используя кнопки Change и Delete изменить назначения.

Probe assignment (Назначение зонда) присваивает легко запоминаемое уникальное имя входу или выходу логической функции. Данное назначение весьма полезно при моделировании системы. Для назначения зонда используют команду Assign/ Probe (рис.2.10)

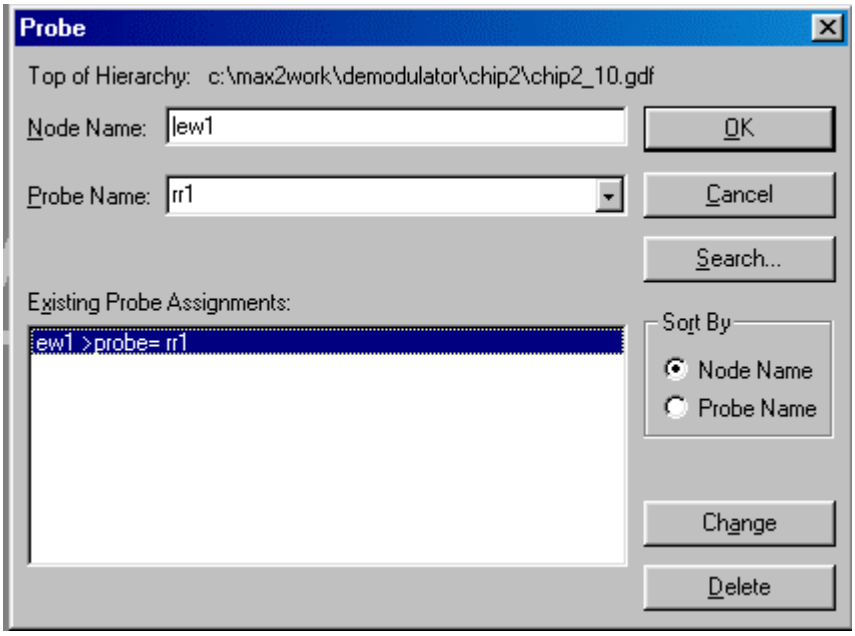

Рис.2.10. Меню команды Assign/ Probe

Connected pin assignment (Назначение соединенных выводов) задает внешнее соединение двух или более выводов на схеме пользователя. Эта информация также полезна в режиме тестирования временных параметров и при тестировании нескольких скомпонованных проектов. Для выполнения назначения соединенных выводов используют команду Assign/ Connected Pins

(рис.2.11)

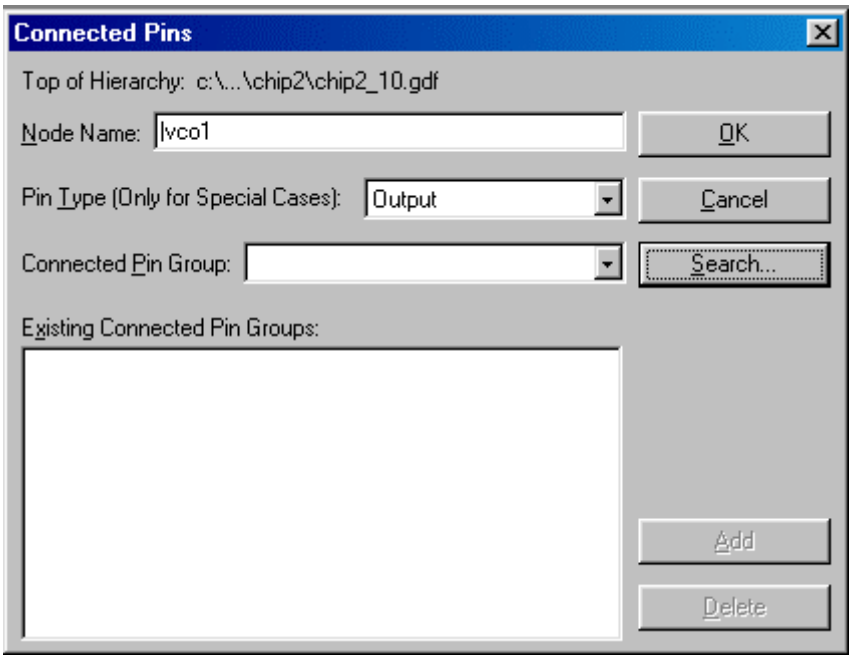

Рис.2.11. Меню команды Assign/ Connected Pins.

Local routing assignment (Назначение местной трассировки) присваивает коэффициент разветвления по выходу узла логическому элементу, находящемуся в том же блоке LAB, что и узел, или же в соседнем LAB, смежным с выбранным узлом, с использованием общих местных связей. Местная трассировка так же производиться между узлом, помещенным в блок LAB на периферии устройства, и выходным контактом, с который он соединен. Назначение местной трассировки производиться с помощью команды **Assign/ Local routing** (рис.2.12)

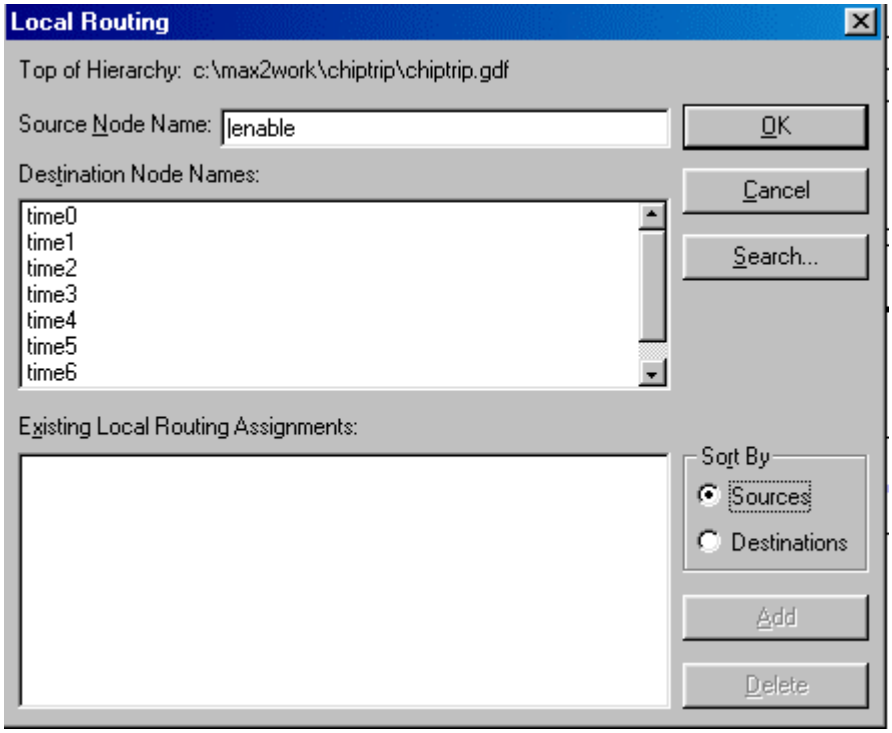

Рис.2.12. Окно команды Assign/ Local routing

Device assignment (Назначение устройства) назначает тип ПЛИС, в которой будет воплощен проект. Если проект состоит из нескольких устройств данный тип назначения делает назначения чипов конкретным устройствам. Можно также выбрать опцию AUTO и предоставить компилятору выбрать устройство из заданного семейства устройств. Процессом

автоматического выбора устройства можно управлять, задавая диапазон и число устройств в семействе. Если проект оказался слишком большим для реализации в одном устройстве, можно задать тип и число дополнительных устройств. Для выбора устройства используется команда **Assign/ Device** (рис.2.13)

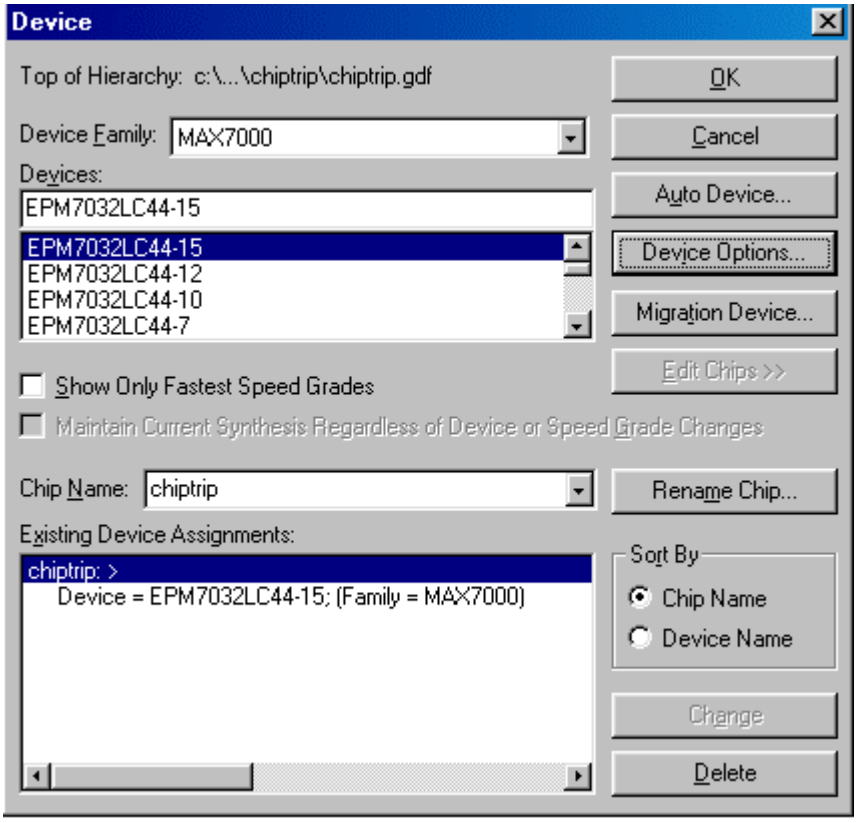

Рис.2.13. Окно команды Assign/ Device

Logic option assignment (Назначение логической опции) управляет логическим синтезом отдельных логических функций во время компиляции с применением стиля логического синтеза и/или отдельных опций логического синтеза. Фирма Altera обеспечивает большое количество логических опций, а также готовых стилей, каждый из которых представляет собой собрание установок для логических опций, объединенное одним именем стиля синтеза (Synthesis style). Пользователь может применять готовые стили или создавать новые. Стили синтеза позволяют настраивать опции синтеза на определенное семейство устройств, учитывая при этом архитектуру семейства. Для настройки стилей синтеза применяется команда **Assign/ Logic Options** (рис.2.14)

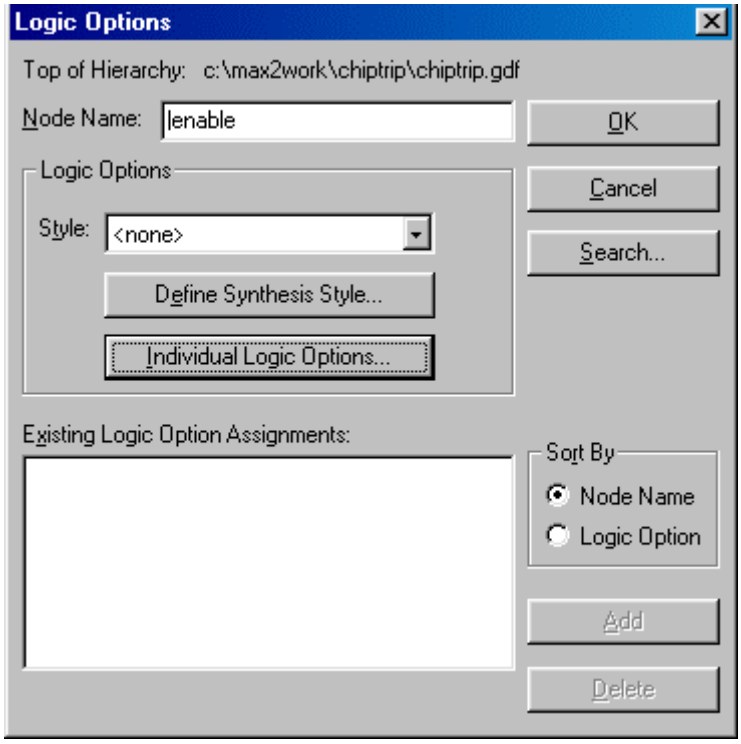

Рис.2.14. Окно команды Assign/ Logic Options

Timing assignment (Назначение временных праметров) управляет логическим синтезом и подгонкой отдельных логических функций для получения требуемых характеристик для времени задержки $\mathbf{t_{PD}}$ (вход-неподрегистренный выход), $\mathbf{t_{CO}}$ (синхросигнал-выход), $\mathbf{t_{SU}}$ (синхросигнал-время установки), $\mathbf{f_{MAX}}$ (частота синхросигнала). Пользователь может также вырезать соединения между путями распространения для конкретного сигнала (называемого "узлом" в системе MAX+PLUS II) и другими узлами проекта. Назначение временных параметров узла производиться по команде **Assign/ Timing Requirements** (рис.2.15)

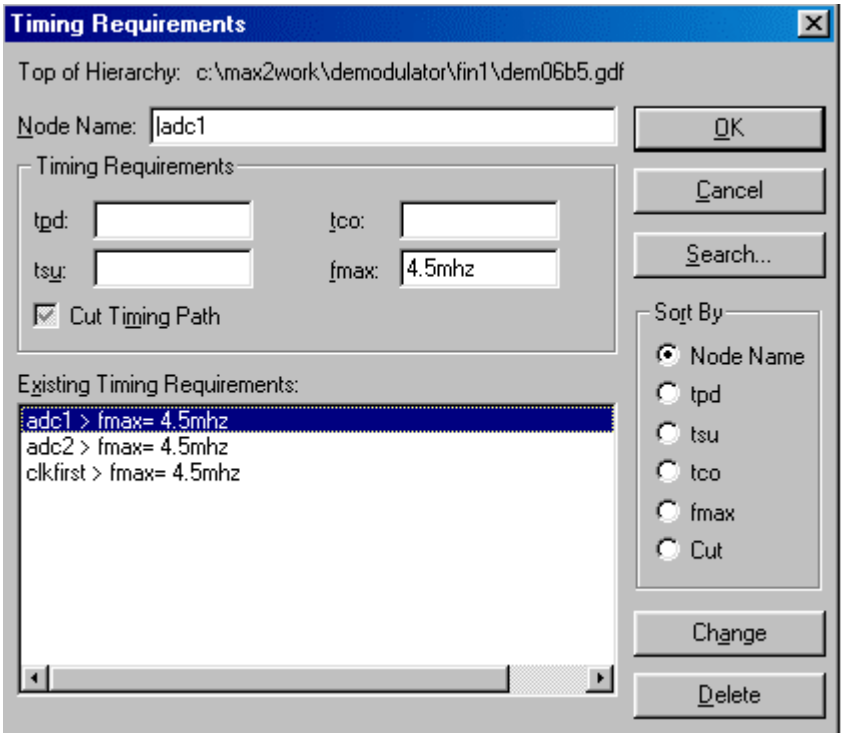

Рис.2.15. Окно команды Assign/ Timing Requirements

Кроме использования команд меню **Assign** назначения можно выполнять щелчком правой кнопки мыши по выбранному

узлу проекта и выбирая соответствующее назначение во всплывающем меню (рис.2.16)

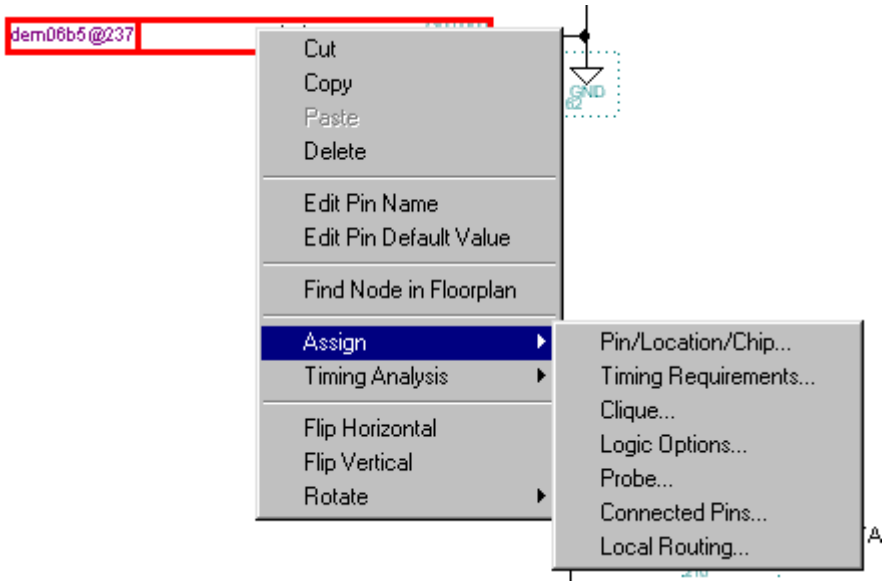

Рис.2.16.

Можно определить глобальные опции устройства для компилятора для того, чтобы он их использовал для всех устройств при обработке проекта. Для резервирования дополнительных возможностей логики на будущее можно задать процентное соотношение выводов и логических элементов, которые должны оставаться неиспользованными во время текущей компиляции. Можно также задать установки для битов опций устройств и выводов в конфигурации устройств, используемой для нескольких целей. Например, можно задать бит защиты от несанкционированного считывания (Security Bit) по умолчанию глобальным, что предотвратит пиратское копирование топологии устройств, базирующихся на памяти EPROM или EEPROM.

Можно задать имена и глобальные установки, которые будут использованы компилятором для параметров во всех параметризованных функций в проекте.

Можно ввести глобальные временные требования для проекта, задавая общие характеристики для времени задержки t_{PD} (вход-нерегистрируемый выход), t_{CO} (синхросигнал-выход), t_{SU} (синхросигнал-время установки), f_{MAX} (частота синхросигнала). Можно также вырезать соединения между всеми двунаправленными контурами обратной связи, цепями прохождения сигналов Preset (установка 1) и Clear (установка 0) и другими цепями синхронизации в проекте. Для этих целей используется команда **Assign/ Global Project Timing Requirements** (рис.2.17)

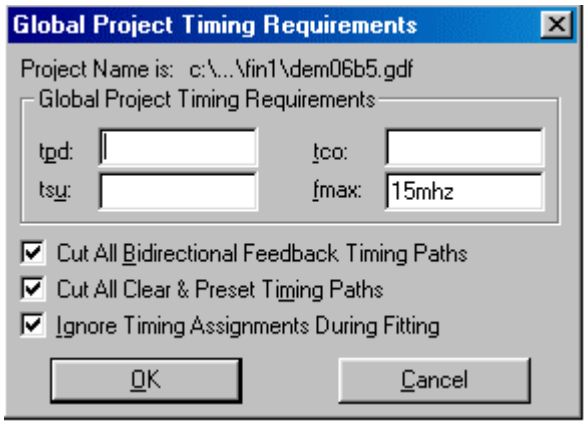

Рис.2.17. Окно команды Assign/ Global Project Timing Requirements

Флажок Cut All Bidirectional Feedback Timing Paths позволяет исключит все цепи обратной связи для двунаправленных выводов

Флажок Cut All Clear & Preset Timing Paths позволяет удалить соединения между всеми цепями сброса и предустановки проекта.

Флажок Ignore Timing Assignments During Fitting позволяет запустить трассировщик (Fitter) без учета временных ограничений проекта. Когда этот флажок сброшен и заданы временные параметры, то осуществляется т.н. управляемый синтез (time driving synthesis).

Из собственного опыта заметим, что в начале для ускорения компиляции следует выбрать этот флажок, а в дальнейшем при “вылизывании” проекта сбросить. Следует помнить, что управляемый временной синтез возможен только для устройств FLEX, для устройств MAX времена задержек предопределены и в слчае назначения временных параметров происходит только лишь проверка соответствия полученных при синтезе параметров заданным.

Можно сделать глобальные установки для компилятора в части логического синтеза проекта. Можно задать используемый по умолчанию стиль логического синтеза, определить степень оптимизации по скорости – занимаемым ресурсам, дать указания компилятору по выбору автоматических глобальных сигналов управления, таких как Clock (тактовый), Clear (установка 0), Preset (установка 1) и Output Enable (разрешение выхода). Можно также выбрать для компилятора режим стандартного или многоуровневого синтеза, режим кодирования цифрового автомата с 1 при подключении питания, а также режим автоматической упаковки регистров. Кроме того, можно выбрать вариант автоматической реализации логики в быстрых входных или выходных логических элементах и ячейках входа/выхода (Slew rate), выводах с открытого стоком (open drain pins) и блоках ячеек памяти EAB. Для назначения глобальных параметров логического синтеза используют команду **Assign/ Global Project Logic Synthesis** (рис.2.18)

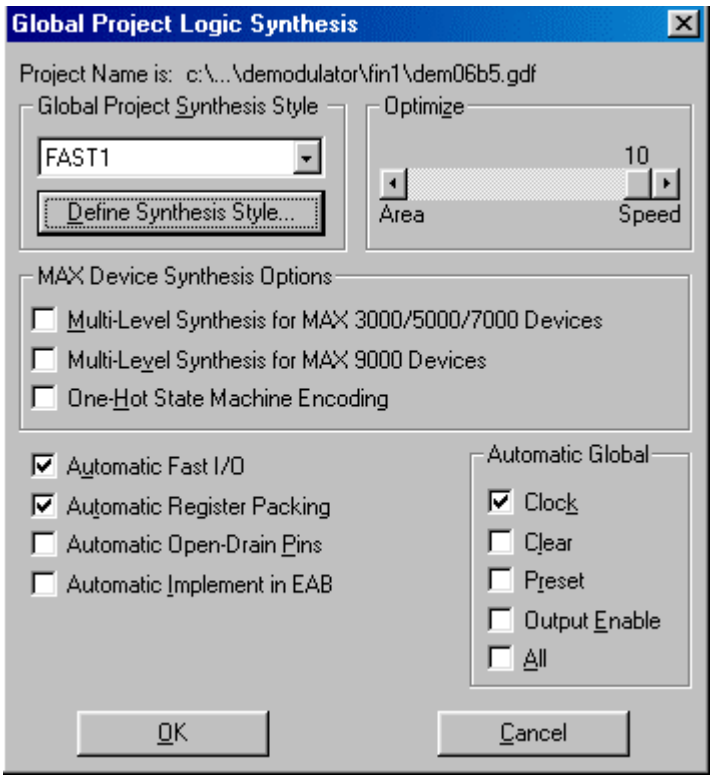

Рис.2.18. Окно команды Assign/ Global Project Logic Synthesis.

Кнопка Define Synthesis Style позволяет выбрать более тонкие параметры стиля синтеза (рис.2.19)

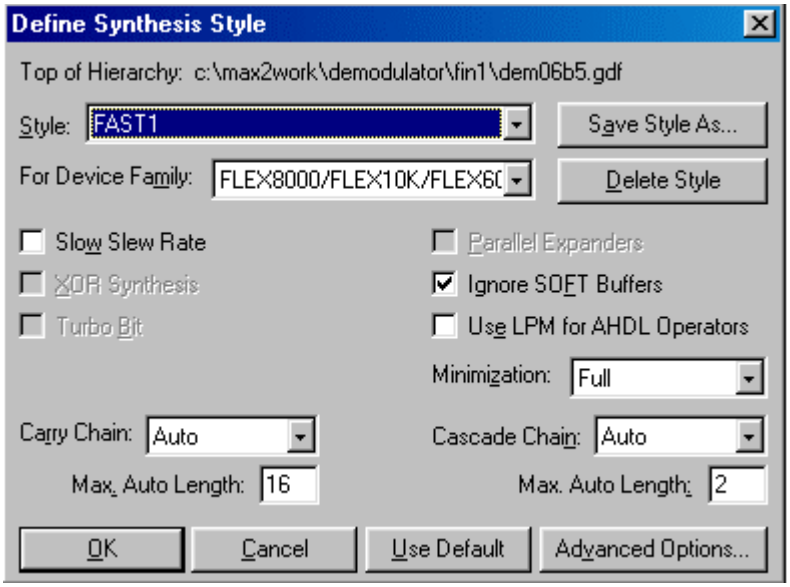

Рис.2.19. Определение стиля синтеза

Как видно из рис.2.19 возможно выбрать имя стиля, способ реализации и максимальную длину цепочек переноса и каскадирования, степень минимизации логических функций и другие параметры синтеза.

Кнопка Advanced Options позволяет выбрать параметры синтеза в диалоговом окне, приведенном на рис.2.20

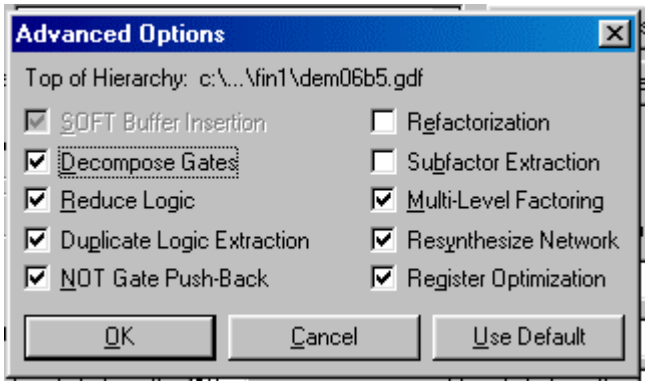

Рис.2.20. Окно Advanced Options.

2.3. Редакторы MAX PLUS II

Все пять редакторов MAX PLUS II и три редактора создания файла проекта (графический, текстовый и сигнальный) имеют общие функции, такие как, например, сохранение и вызов файла. Кроме того, приложения редактора MAX PLUS II имеют следующие общие функции:

создание файлов символов и файлов с прототипами функций (Include-файлы)(symbol and include file generation);

поиск узлов (node location);

траверз иерархического дерева(hierarchy traversal);

всплывающие окна меню, зависящего от контекста (context-sensitive menu comands);

временной анализ(Timing Analysis);

поиск и замена фрагментов текста (Find & Replace Text);

отмена последнего шага редактирования, его возвращения, вырезка, копирование, вклеивание и удаление выбранных фрагментов, обмен фрагментами между приложениями MAX PLUS II или приложениями Windows (Undo, Cut, copy,Paste & Delete);

печать (print).

На рис. 2.21 показано окно графического редактора(Graphic Editor) MAX PLUS II, обеспечивающего проектирование в реальном формате изображения (WYSIWIG). В нем можно создавать новые файлы (команда **New** из меню **File**). Вызывается графический редактор из меню **MAX PLUS II**.

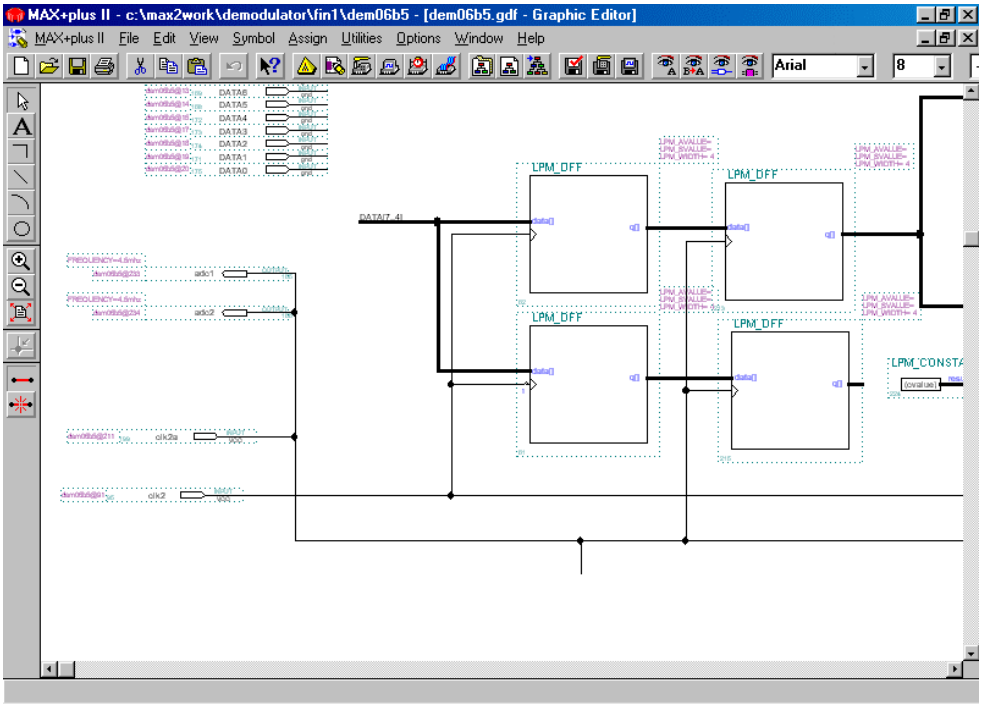

Рис. 2.21. Графический редактор MAX PLUS II

Графические файлы проекта (**.gdf**) или схемные файлы OrCAD (**.sch**), созданные в данном графическом редакторе, могут включать любую комбинацию символов примитивов, мегафункций и макрофункций. Символы могут представлять собой любой тип файла проекта (**.gdf .sch .tdf .vhd .v .wdf .edf .xnf .adf .smf**). Универсальность графического редактора характеризуется следующими чертами:

Инструмент выбора (“стрелка”) облегчает разработку проекта. Он позволяет двигать и копировать объекты, а также вводить новые символы. Когда вы помещаете его на вывод или конец линии, он автоматически преобразуется в инструмент рисования ортогональных линий. Если им щелкнуть на тексте, он автоматически преобразуется в инструмент редактирования текста;

Символы соединяются сигнальными линиями, которые называют узлами (nodes), или линиями шин (bus), которые представляют собой несколько логически сгруппированных узлов. Когда вы присваиваете узлу имя, вы можете соединить его с другими узлами или символами только по имени. Шины соединяются по имени, но возможно и их графическое соединение;

Пользователь может переопределить порты, используемые в каждом отдельном примере символа мега- или макрофункции, а также инвертировать их. При этом для указания инвертированного порта появится кружок, обозначающий инверсию;

Можно выбрать несколько объектов в прямоугольной области и редактировать их вместе или по отдельности. При перемещении выбранной области сигнальные связи сохраняются;

Для каждого символа можно просматривать назначения зондов, выводов, расположения, чипов, клик, временных параметров, местную трассировку, логические опции и назначения параметров. Для облегчения тестирования можно также создать назначения групп выводов, которые будут определять соединения внешнего устройства между выводами;

Поставляемые фирмой Altera примитивы, мега- и макрофункции сокращают время разработки дизайна. Пользователь может

также создавать свои собственные библиотеки функций. При редактировании символа или восстановлении его по умолчанию можно автоматически создавать выбранные примеры или все примеры этого символа в файле в графическом редакторе.

Графический редактор обеспечивает и много других возможностей. Например, можно увеличить или уменьшить масштаб отображения на экране и увидеть дизайн целиком или какую-либо его деталь. Можно выбирать гарнитуру и размер шрифта, задавать стили линий, устанавливать и отображать направляющие. Можно копировать, вырезать, вклеивать и удалять выбранные фрагменты; получать зеркальное отображение (вертикальное или горизонтальное; поворачивать выделенные фрагменты на 90, 180 или 270 градусов; задавать размер, рамещение текущего листа схемы по вертикали или горизонтали.

На рис. 2.22 представлено окно символьного редактора системы MAX PLUS II, с помощью которого можно просматривать, создавать и редактировать символ, представляющий собой логическую схему. В нем можно создавать новые файлы (команда **New** из меню **File**). Вызывается символьный редактор из меню **MAX PLUS II**.

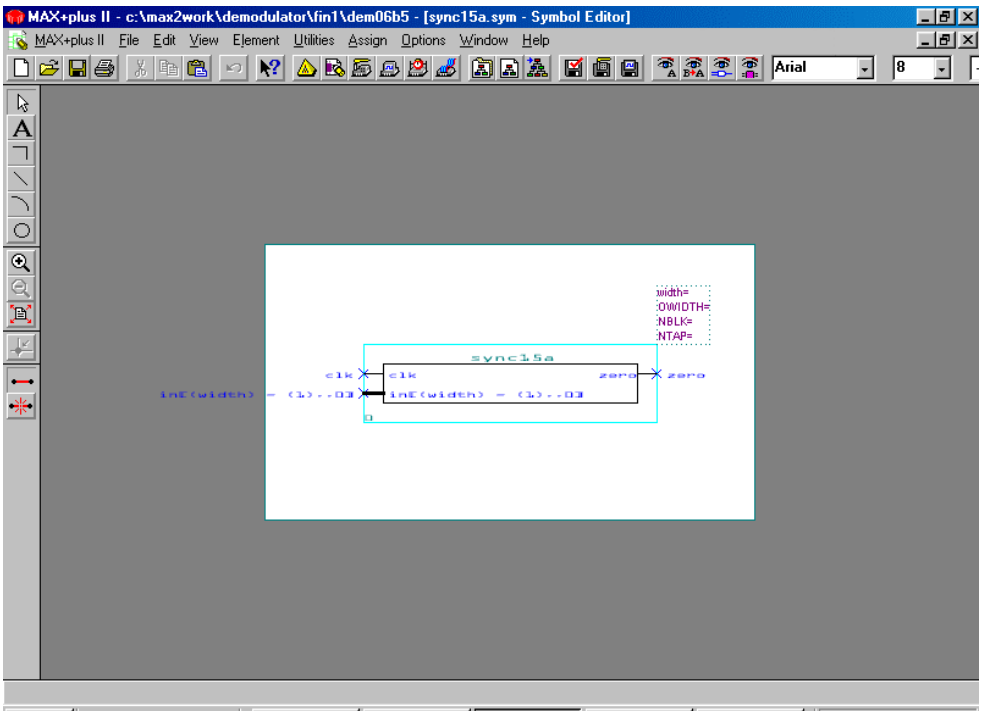

Рис. 2.22. Символьный редактор MAX PLUS II

Символьный файл имеет то же имя, что и файл проекта, и расширение **.sym**. Команда **Creat Default Suymbol** меню **File**, которая есть в графическом, текстовом и сигнальном редакторах, создает символ для любого файла дизайна. Символьный редактор обладает следующими характеристиками:

- можно переопределить символ, представляющий файл проекта;

- можно создавать и редактировать выводы и их имена, разрабатывая входные, выходные и двунаправленные контакты, а также задавать варианты ввода символа в файл графического редактора: с отображением на экране имен выводов или без отображения, с отображением полного или сокращенного имени. Таким образом, полное имя порта и имя, отображаемое в файле в окне графического редактор, могут быть разными;

- имена выводов автоматически дублируются за границу символа. Редактированию подлежат только имена внутри границы символа Имена снаружи нельзя менять, они просто иллюстрируют соединение выводов;

- можно задать значения параметров и их значения по умолчанию;

- сетка и направляющие помогают выполнить точное выравнивание объектов;

- в символе можно вводить комментарии или полезные замечания, которые также появятся при вводе символа в файл в графическом редакторе.

На рис. 2.23 показано окно текстового редактора MAX PLUS II, который является гибким инструментом для создания текстовых файлов проекта на языках описания аппаратуры: AHDL (**.tdf**), VHDL (**.vhd**), Verilog HDL (**.v**). В этом текстовом редакторе можно работать также с произвольным файлом формата ASCII. В нем можно создавать новые файлы (команда **New** из меню **File**). Вызывается символьный редактор из меню **MAX PLUS II**.

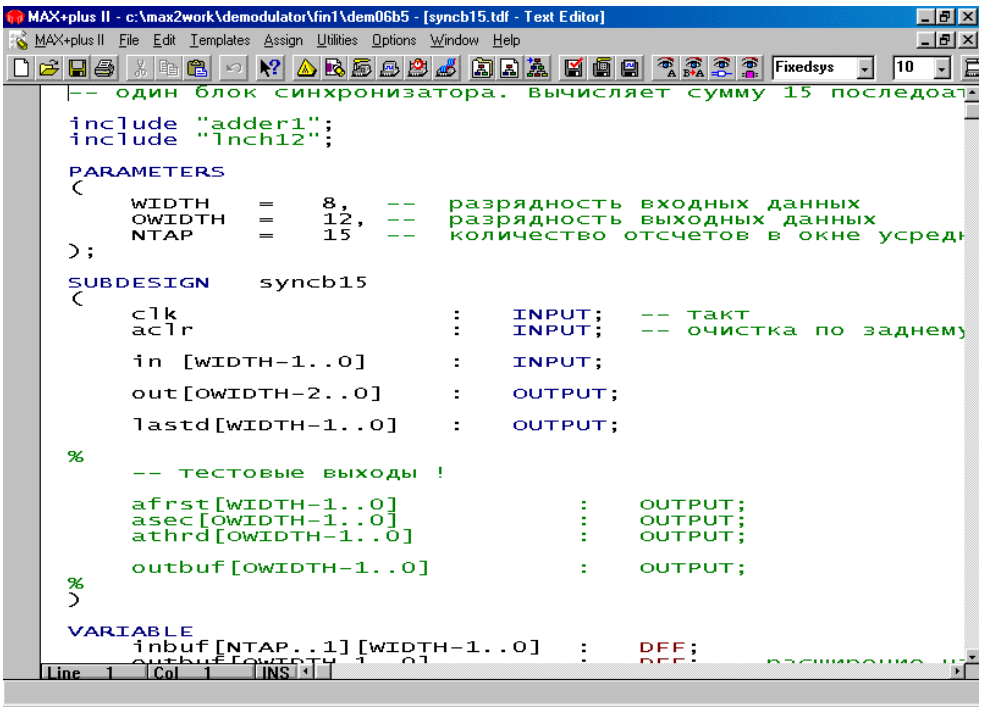

Рис. 2.23. Текстовый редактор MAX PLUS II

Все перечисленные файлы проекта можно создавать в любом текстовом редакторе, однако данный редактор имеет встроенные возможности удобного ввода файлов проекта, их компиляции и отладки с выдачей сообщений об ошибках и их локализацией в исходном тексте или в тексте вспомогательных файлов; кроме того, существуют шаблоны языковых конструкций для AHDL, VHDL и Verilog HDL, выполнено окрашивание синтаксических конструкций. В данном редакторе можно вручную редактировать файлы назначений и конфигурации (**.acf**), а также делать установки конфигурации для компилятора, симулятора и временного анализатора .

Пользуясь данным текстовым редактором, можно создавать тестовые векторы (**.vec**), используемые для тестирования, отладки функций и при вводе сигнального проекта. Можно также создавать командные файлы (**.cmd** – для симулятора и **.edc** – для EDIF), а также макробиблиотеки (**.lmf**).

В текстовом редакторе MAX PLUS II обеспечивается контекстуальная справка.

Сигнальный редактор (Waveform Editor) (рис. 2.24) выполняет две роли: служит инструментом создания дизайна и инструментом ввода тестовых векторов и просмотра результатов тестирования. Пользователь может создавать сигнальные файлы проекта (**.wdf**), которые содержат логику дизайна для проекта, а также файлы каналов тестирования (**.scf**), которые содержат входные вектора для тестирования и функциональной отладки. Новый файл создается командой **New** меню **File**. Вызывается сигнальный редактор из меню **MAX PLUS II**.

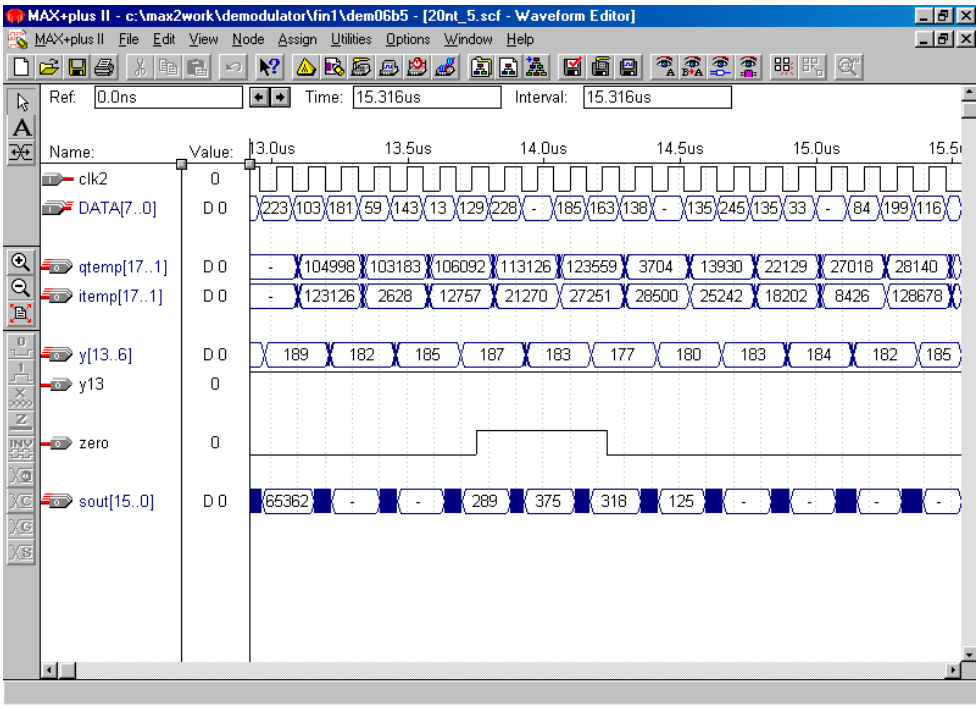

Рис. 2.24. Сигнальный редактор MAX PLUS II

Разработка дизайна в сигнальном редакторе является альтернативой созданию дизайна в графическом или текстовом редакторах. Здесь можно графическим способом задавать комбинации входных логических уровней и требуемых выходов. Созданный таким образом файл с расширением WDF (Waveform design file) может содержать как логические входы, так и входы цифрового автомата, а также выходы комбинаторной логики, счетчиков и цифровых автоматов. Можно использовать также “замурованные” (buried) узлы, которые помогают определить требуемые выходы.

Способ разработки проекта в сигнальном редакторе лучше подходит для цепей с четко определенными последовательными входами и выходами, т.е. для цифровых автоматов, счетчиков и регистров.

С помощью сигнального редактора можно легко преобразовывать формы сигналов целиком или частично, создавая и редактируя узлы и группы. Простыми командами можно создавать файл таблицы ASCII-символов (**.tbl**) или импортировать файл тестовых векторов в формате ASCII (**.vec**) для создания файлов тестируемых каналов SCF и сигнального дизайна WDF. Можно также сохранить файл WDF как SCF для проведения тестирования или преобразовать SCF в WDF для использования его в качестве файла проекта.

Сигнальный редактор имеет следующие отличительные черты:

- можно создать или отредактировать узел для получения типа I/O (вход/выход), который представляет собой входной или выходной контакт или “замурованную” логику;

- при разработке WDF можно задать тип логики, которая делает каждый узел контактом, причем входным, регистровым, комбинаторным или цифровым автоматом;

- можно также задать значения по умолчанию в логическом узле для активного логического уровня: высокий (1), неопределенный (X) или с высоким импедансом (Z), а также имя состояния по умолчанию в узле типа цифрового автомата;

- для упрощения создания тестового вектора можно легко добавить в файл тестируемых каналов SCF несколько узлов или все из информационного файла симулятора (**.snf**), существующего для полностью откомпилированного и оптимизированного проекта;

- можно объединять от 2 до 256 узлов для создания новой группы (шины) или разгруппировывать объединенные ранее в группу узлы. Можно также объединять группы с другими группами. Значение группы может быть отображено в двоичной, десятичной, шестнадцатеричной или восьмеричной системе счисления с преобразованием (или без) в код Грэя;

- можно копировать, вклеивать, перемещать или удалять выбранную часть (“интервал”) формы сигнала или всю форму сигнала, а также весь узел или группу (т.е. имя узла или группы плюс форму сигнала). Одной операцией можно отредактировать несколько интервалов, целые формы сигналов, а также целые узлы и группы. Копии целых узлов и групп связаны, так что редакционные правки одной копии отражаются во всех копиях. Можно также инвертировать, вставлять, переписывать, повторять, расширять или сжимать интервал формы сигнала любой длины с любым логическим уровнем, тактовым сигналом, последовательностью счета или именем состояния;

- можно задать и, по желанию, отображать на экране сетку для выравнивания переходов между логическими уровнями либо до их создания, либо после;

- в любом месте файла можно вводить комментарии между формами сигнала;

- можно менять масштаб отображения;

- для того, чтобы показать разницу между выходами при тестировании и выходами реального устройства, можно сделать наложение любых выходов в текущем файле или наложить второй файл сигнального редактора для сравнения форм сигналов его узлов и групп с соответствующими изи текущего файла.

Для отладки устройств ЦОС часто приходится тестировать алгоритм на реальных или смоделированных сигналах. Для этого удобно использовать векторныйс сигнальный файл (Vector File)

Vector File (формат текста ASCII) используется для определения входных условий моделирования и узлов, которые нужно моделировать. Vector File может также использоваться, чтобы создать Waveform Design File для входных данных проекта. Рассмотрим формат векторного файла подробнее.

Все разделы, используемые в Vector File рассматриваются ниже в том порядке, в котором они обычно присутствуют в файле. Возможноь повторение любого раздела с целью внесения дополнительных условий для входных данных в пределах одного Vector File.

Unit Section

Начинается с ключевого слова **UNIT** с дальнейшим указанием единиц измерения в файле. Параметр необязательный. По умолчанию единицы измерения ns. Возможные единицы измерения: ns(нс), ms(мс), us(мкс), s(с), mhz(МГц). Раздел заканчивается символом ; .

Пример: UNIT ms;

Start Section

Начинается с ключевого слова **START** с последующим указанием начального временного значения. Параметр необязательный. Значение по умолчанию нулевое. Если не указаны единицы измерения, то они принимаются из раздела Unit Section. Раздел заканчивается символом ; .

Пример: START 5ns;

Stop Section

Аналогичен разделу Start Section. По умолчанию принимается значение времени последнего вектора модели.

Пример: STOP 150ms;

Необходимо учитывать, что Vector File должен содержать кратное количество Start-Stop Section, представляющих собой временные интервалы. Недопустимо обращение к одному временному интервалу различных векторов модели.

Interval Section

Начинается с ключевого слова **INTERVAL** с последующим указанием временного значения. Определяет временной интервал

ввода векторов. Параметр необязательный. Значение по умолчанию 1 нс. Раздел заканчивается символом ; .

Пример: INTERVAL 15ns;

Group Create Section

Начинается с ключевого слова **GROUP CREATE**. Данный раздел требуется не всегда для групп, шин, или конечных автоматов, которые были созданы в исходных файлах проекта. Все узлы в группе должны иметь тип I/O. В Vector File, используемом для симуляции, узлы должны иметь имена, совпадающие с именами узлов, заведенными в файле-проекте, включая иерархический путь, если это необходимо. Раздел заканчивается символом ; .

Пример: GROUP CREATE groupABC = nodeA nodeB nodeC;

Radix Section

Начинается с ключевого слова **RADIX** с последующим указанием обозначения системы счисления. Параметр необязательный. По умолчанию принимается шестнадцатиричная система счисления. Различают четыре системы: BIN(двоичная), DEC(десятичная), HEX(шестнадцатиричная), OCT(восьмеричная). Раздел заканчивается символом ; .

Пример: RADIX DEC;

Inputs Section

Начинается с ключевого слова **INPUTS**. Далее следует список узлов и/или имен групп. В Vector File, используемом для симуляции, имена узлов должны совпадать с именами узлов в файле-проекте, включая иерархический путь, если это необходимо.

Пример: INPUT databus clk OEN nodeA;

Раздел заканчивается символом ; .

В приведенном ниже примере при задействовании следующей секции входных данных значения в предыдущей секции теряются.

Пример: INPUTS A1 A2;

```
START 0;

STOP 25;

PATTERN %Секция модели 1 со входами A1 и A2%

0 0 0

0 0 1;

START 26;

STOP 50;

PATTERN %Секция модели 2 со входами A1 и A2%

0 1 0

1 0 0;
```

```
INPUTS A1 B1;

START 51;

STOP 100;

PATTERN %Секция модели 3 со входами A1 и B1%

1 1 0

0 1 1;
```

Outputs Section

Начинается с ключевого слова **OUTPUTS**. Используется для описания выходов. Аналогия с Inputs Section.

Пример: OUTPUTS RCO QA QB QC;

Пример: OUTPUTS A1 A2;

```
START 0;

STOP 25;

PATTERN %Секция модели 1 с выходами A1 и A2%

0 0 0

0 0 1;

START 26;

STOP 50;

PATTERN %Секция модели 2 с выходами A1 и A2%

0 1 0

1 0 0;

OUTPUTS A1 B1;

START 51;

STOP 100;

PATTERN %Секция модели 3 с выходами A1 и B1%

1 1 0

0 1 1;
```

Buried Section

Начинается с ключевого слова **BURIED**. Используется для описания узлов. Аналогия с Inputs Section.

Пример: BURIED nodeQA0 nodeQA1 nodeQB0 nodeQB1;

Pattern Section

Начинается с ключевого слова PATTERN. В этом разделе используются данные предыдущих разделов Inputs, Outputs и Buried Section. Добавление новых Inputs, Outputs и Buried Section очищает все предыдущие данные этих типов, т.е. старые данные заменяются новыми (см. пример для Inputs Section). Раздел заканчивается символом ; .

Vector File, используемый, чтобы создать WDF может содержать дополнительные разделы **Combinatorial Section, Machine Section, Registered Section.** Эти разделы игнорируются, если Vector File используется для моделирования.

Рассмотрим пример создания Vector File для восьмибитного сумматора.

Описание на AHDL

sum8_.tdf

```
SUBDESIGN SUM8_

%8 разрядный сумматор без знака%

(

clk:INPUT;

a[7..0]:INPUT = GND;

b[7..0]:INPUT = GND;

sum[7..0]:OUTPUT;

cr:OUTPUT;

)

VARIABLE

c[7..1]:NODE;

sum[7..0]:DFF;

cr:DFF;

BEGIN

sum[7..0].clk=clk;

sum[7..0].prn=VCC;

sum[7..0].clrn=VCC;

cr.clk=clk;

cr.prn=VCC;

cr.clrn=VCC;
```

```
If (a[0]&b[0])==VCC then

sum[0]=GND;

c[1]=VCC;

ElsIf (a[0]#b[0])==GND then

sum[0]=GND;

c[1]=GND;

Else sum[0]=VCC;

c[1]=GND;

End If;

%%

FOR i IN 1 TO 6 GENERATE

If (a[i]&b[i]&c[i])==VCC then

sum[i]=VCC;

c[i+1]=VCC;

ElsIf (a[i]#b[i]#c[i])==GND then

sum[i]=GND;

c[i+1]=GND;

ElsIf (((a[i]&b[i])#(a[i]&c[i])#(b[i]&c[i]))&!(a[i]&b[i]&c[i]))==VCC then

sum[i]=GND;

c[i+1]=VCC;

Else sum[i]=VCC;

c[i+1]=GND;

End If;

END GENERATE;

%%

If (a[7]&b[7]&c[7])==VCC then

sum[7]=VCC;
```

```
cr=VCC;
ElsIf (a[7]#b[7]#c[7])==GND then
sum[7]=GND;
cr=GND;
ElsIf (((a[7]&b[7])#(a[7]&c[7])#(b[7]&c[7]))&!(a[7]&b[7]&c[7]))==VCC then
sum[7]=GND;
cr=VCC;
Else sum[7]=VCC;
cr=GND;
End If;
END;
```

sum8_.vec

```
START 0;
STOP 500;
INTERVAL 25;
RADIX BIN;
INPUTS clk;
PATTERN
0 1;
START 0;
STOP 500;
INTERVAL 50;
RADIX BIN;
INPUTS a7 a6 a5 a4 a3 a2 a1 a0 b7 b6 b5 b4 b3 b2 b1 b0;
OUTPUTS sum[7..0] cr;
PATTERN
%a7 a6 a5 a4 a3 a2 a1 a0 b7 b6 b5 b4 b3 b2 b1 b0%
```

```
1 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1

1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1

0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0

1 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
```

В данном примере будут сложены 6 пар двоичных чисел. При этом будет задан тактирующий меандр clk (тактирующий импульс для D-триггеров) с периодом повторения 50нс и скважностью 2. Начальным значением импульса будет 0. Преобразование будет проводиться от 0нс до 500нс.

При начале симуляции необходимо отсутствие в рабочей директории соответствующего файла sum8_.scf, т.к. иначе симулятор будет по умолчанию использовать файл не с расширением .vec, а файл с расширением .scf. После процесса симуляции с использованием .vec файла файл с расширением .scf будет создан автоматически.

На рис. 2.25 показано окно поуровневого планировщика (FloorPlan Editor), с помощью которого пользователь назначает ресурсы физических устройств и просматривает результаты разветвлений и монтажа, сделанных компилятором. Окно поуровневого планировщика открывается при выборе опции **Floorplan Editor** в меню **MAX PLUS II**.

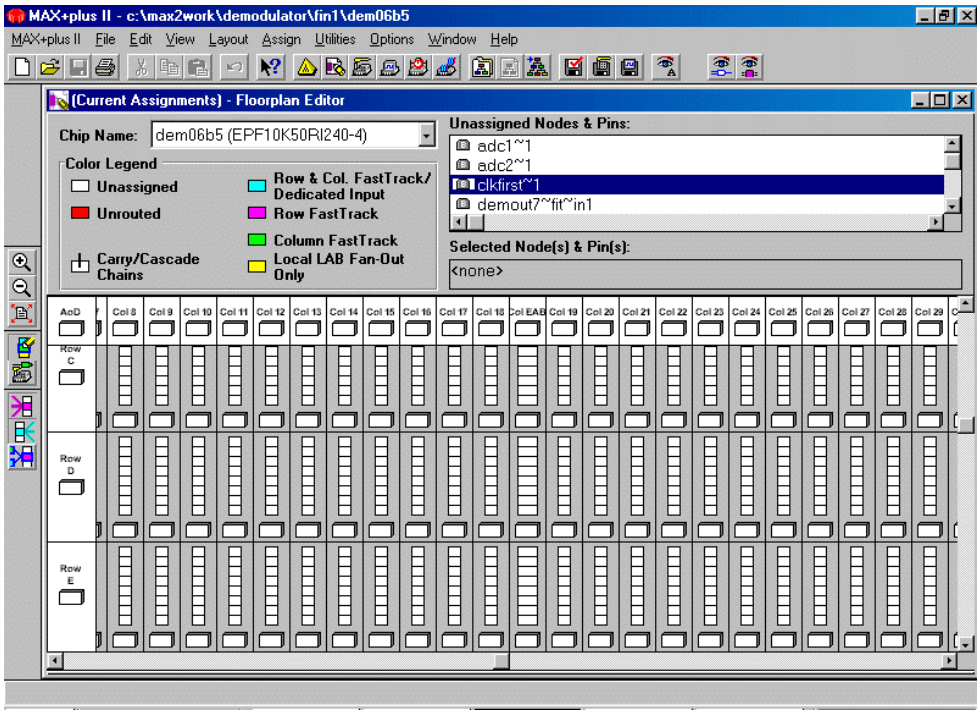

Рис. 2.25. Поуровневый планировщик MAX PLUS II

В окне поуровневого планировщика могут быть представлены два типа изображения:

Device View (Вид устройства) показывает все выводы устройства в сборке и их функции;

LAB View (Вид логического блока) показывает внутреннюю структуру устройства, в том числе все логические блоки (LAB) и отдельные логические элементы или макроячейки.

2.4. Процесс компиляции

Сначала компилятор извлекает информацию об иерархических связях между файлами проекта и проверяет проект на простые

ошибки ввода дизайнов. Он создает организационную карту проекта и затем, комбинируя все файлы проекта, превращает их в базу данных без иерархии, которую может эффективно обрабатывать.

Компилятор применяет разнообразные способы увеличения эффективности проекта и минимизации использования ресурсов устройства. Если проект слишком большой, чтобы быть реализованным в одной ПЛИС, компилятор может автоматически разбить его на части для реализации в нескольких устройствах того же самого семейства ПЛИС, при этом минимизируется число соединений между устройствами. В файле отчета (.**rpt**) затем будет отражено, как проект будет реализован в одном или нескольких устройствах.

Кроме того, компилятор создает файлы программирования или загрузки, которые программатор системы MAX PLUS II или другой будет использовать для программирования одной или нескольких ПЛИС фирмы Altera.

Несмотря на то, что компилятор может автоматически компилировать проект, существует возможность задать обработку проекта в соответствии с точными указаниями разработчика. Например, возможно задать стиль логического синтеза проекта по умолчанию и другие параметры логического синтеза в рамках всего проекта. Кроме того, удобно задать временные требования в рамках всего проекта, точно задать разбиение большого проекта на части для реализации в нескольких устройствах и выбрать варианты параметров устройств, которые будут применены для всего проекта в целом. Вы можете также выбрать, сколько выводов и логических элементов должно быть оставлено неиспользованными во время текущей компиляции, чтобы зарезервировать их для последующих модификаций проекта.

Компиляцию можно запустить из любого приложения MAX PLUS II или из окна компилятора. Компилятор автоматически обрабатывает все входные файлы текущего проекта. Процесс компиляции можно наблюдать в окне компилятора в следующем виде (рис.2.26):

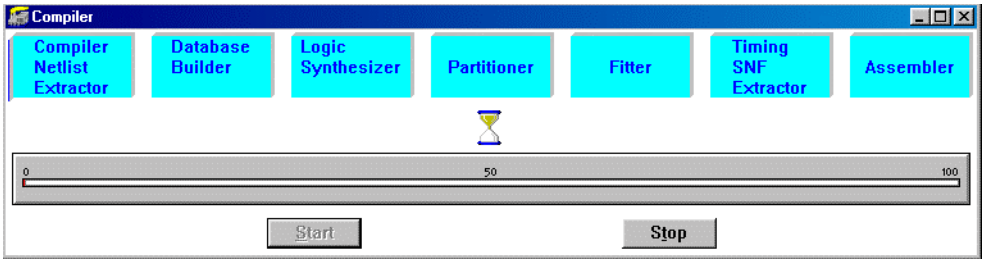

Рис.2.26. Процесс компиляции

- опустошаются и переворачиваются песочные часы, что указывает на активность компилятора;

- высвечиваются прямоугольники модулей компилятора по очереди, по мере того как компилятор завершает каждый этап обработки;

- под прямоугольником модуля компилятора появляется пиктограмма выходного файла, сгенерированного данным модулем. Для открытия соответствующего файл а следует дважды щелкнуть левой кнопкой мыши на пиктограмме, и откроется;

- процент завершения компиляции постепенно увеличивается (до 100%), что отражается также растущим прямоугольником “градусник”;

- во время разбиения и монтажа кнопка компилятора **Stop** (Стоп) превращается в кнопку **Stop/Show Status** (Стоп/Показать состояние), которую вы можете выбрать для открытия диалогового окна, в котором отражается текущее состояние разбиения и монтажа проекта;

- при обнаружении в процессе компиляции каких-либо ошибок или возможных проблем автоматически открывается окно обработчика сообщений, в котором отображается список сообщений об ошибке, предупреждающих и информационных сообщений, а также сразу дается справка по исправлению ошибки. Кроме того, вы можете определить источники сообщений в файлах проекта или в его поуровневом плане назначений.

Компилятор может работать в фоновом режиме. Вы можете уменьшить до минимума окно компилятора, пока он обрабатывает проект, и продолжить работу над другими файлами проекта. Растущий прямоугольник “градусник” под пиктограммой уменьшенного окна компилятора позволяет вам наблюдать за продвижением процесса компиляции в то время, как вы можете сосредоточить свое снимание на другой задаче. Однако следует помнить, что такая роскошь, как многозадачная работа, возможна только на приличной машине. Если существует дефицит оперативной памяти, то лцчше во

время процесса компиляции спокойно пить чай.

Компилятор системы MAX PLUS II обрабатывает проект, используя следующие модули и утилиты:

Compiler Netlist Extractor (экстрактор списка цепей), включающий встроенные программы чтения форматов EDIF, VHDL, Verilog и XNF;

Database Builder (построитель базы данных);

Logic Synthesizer (логический синтезатор);

Partitioner (разделитель);

Fitter (трассировщик);

Functional SNF Extractor (экстрактор для функционального тестирования);

Timing SNF Extractor (экстрактор для тестирования временных параметров);

Linked SNF Extractor (экстрактор для тестирования компоновки);

EDIF Netlist Writer (программа записи выходного файла в формат EDIF);

Verilog Netlist Writer (программа записи выходного файла в формат Verilog);

VHDL Netlist Writer VHDL (программа записи выходного файла в формат VHDL);

Assembler (модуль ассемблера);

Design Doctor Utility (утилита диагностики проекта).

Модуль Compiler Netlist Extractor (экстрактор форматов) преобразует каждый файл проекта в один или несколько двоичных файлов с расширением **.cnf**. (compiler netlist file) Поскольку компилятор подставляет значения всех параметров, используемых в параметризованных функциях, содержимое файла CNF может меняться при последовательной компиляции, если значения параметров меняются. Данный модуль создает также файл иерархических взаимосвязей (**.hif**) (hierarchy interconnect file), в котором документируются иерархические связи между файлами проекта, а также содержится информация, необходимая для показа иерархического дерева проекта в окне Hierarchy Display. Кроме того, данный модуль создает файл базы данных узлов (**.ndb**) (node database), в котором содержатся имена узлов проекта для базы данных назначений ресурсов. Встроенные программы чтения форматов EDIF, VHDL, Verilog и XNF автоматически транслируют информацию проекта в файлы соответствующих форматов **.edf**, **.vhd**, **.v**, **.xnf**, в формат, совместимый с системой MAX PLUS II. Программа чтения формата EDIF обрабатывает входные файлы EDIF с помощью библиотечных файлов **.lmf**, (library mapping file) которые устанавливают соответствие между логическими функциями, разработанными в других САПР, и функциями системы MAX PLUS II. Программа чтения формата XNF может создавать файл для экспорта текстового дизайна (**.tdx**)(text design export file), который содержит информацию на языке AHDL, которая эквивалентна той, что содержится в файле формата XNF (**.xnf**); это делается для того, чтобы редактировать проект на языке AHDL.

Модуль Database Builder (построитель базы данных) использует файл иерархических связей HIF для компоновки созданных компилятором файлов CNF, в которых содержится описание проекта. На основании данных об иерархической структуре проекта данный модуль копирует каждый файл CNF в одну базу данных без иерархической структуры. Таким образом, эта база данных сохраняет электрическую связность проекта.

При создании базы данных модуль исследует логическую полноту и согласованность проекта, а также проверяет пограничную связность и наличие синтаксических ошибок (например, узел без источника или места назначения). На этой стадии компиляции обнаруживается большинство ошибок, которые могут быть тут же легко исправлены. Каждый модуль компилятора последовательно обрабатывает и обновляет эту базу данных.

Первый раз, когда компилятор обрабатывает проект, все файлы проекта компилируются. Вы можете использовать возможность “быстрой повторной компиляции” (smart recompile) для создания расширенной базы данных проекта, которая помогает ускорить последующие компиляции. Эта база данных позволяет вам изменить назначения ресурсов физического

устройства, такие как назначения выводов и логических элементов, а также повторно компилировать проект без повторного построения базы данных и повторного синтеза логики проекта. Используя возможность “полной повторной компиляции” (total recompile), возможно сделать выбор между повторной компиляцией только тех файлов, которые редактировались после последней компиляции, и полной повторной компиляцией всего проекта.

Модуль логического синтезатора (**Logic Synthesizer)** применяет ряд алгоритмов, которые уменьшают использование ресурсов и убирают дублированную логику, обеспечивая тем самым максимально эффективное использование структуры логического элемента для архитектуры целевого семейства устройств. Данный модуль компилятора применяет также способы логического синтеза для требований пользователя по временных параметров и др. Кроме того, логический синтезатор ищет логику для несоединенных узлов. Если он находит неприсоединенный узел, он убирает примитивы, относящиеся к этому узлу.

Для управления логическим синтезом имеются три заранее описанных стиля и большое число логических опций.

В любом приложении MAX PLUS II можно ввести значения временных параметров проекта и выбрать логические опции, а также определить стили логического синтеза. Вы можете задать глобальный логический синтез по умолчанию и временные параметры проекта для всего проекта в целом, а также все дополнительные назначения логических опций и временных параметров проекта и для отдельных логических функций.

Если проект не помещается при монтаже в одно устройство, модуль **Partitioner** (разделитель) разделяет базу данных, обновленную логическим синтезатором, на несколько ПЛИС одного и того же семейства, стараясь при этом разделить проект на минимально возможное число устройств. Разбиение проекта происходит по границам логических элементов, а число выводов, используемое для сообщения между устройствами, минимизируется.

Разбиение может быть проведено полностью автоматически либо под частичным управлением со стороны пользователя, либо полностью под управлением со стороны пользователя. Назначения устройств и установки для автоматического выбора устройств позволяют применить тот уровень управления, который наиболее подходит для конкретного проекта.

Когда работают модули Partitioner и Fitter, вы можете приостановить компиляцию. Компилятор отобразит информацию о текущем состоянии процессов разбиения и трассировки кристалла, в том числе сравнение требуемых и имеющихся ресурсов. Это нужно для того, чтобы принять решение, продолжать ли компиляцию, или вносить кардинальные изменения в проект.

Используя базу данных, обновленную модулем разбиения, модуль трассировки (Fitter) приводит в соответствие требования проекта с известными ресурсами одного или нескольких устройств. Он назначает каждой логической функции расположение реализующего ее логического элемента и выбирает соответствующие пути взаимных соединений и назначения выводов. Данный модуль пытается согласовать назначения ресурсов, т.е. выводов, логических элементов, элементов ввода/вывода, ячеек памяти, чипов, клик, устройств, местной трассировки, временных параметров и назначения соединенных выводов из файла назначений и конфигурации (**.acf**) (Assignment & Configuration file), с имеющимися ресурсами. Модуль имеет параметры, позволяющие определить способы трассировки, например, автоматическое введение логических элементов или ограничение коэффициента объединения по входу. Если трассировка не может быть выполнена, модуль выдает сообщение и предлагает вам выбор, проигнорировать некоторые или все ваши назначения либо прекратить компиляцию.

Независимо от того, завершена ли полная трассировка проекта, данный модуль генерирует файл отчета (**.rpt**) (report file), в котором документируется информация о разбиении проекта, именах входных и выходных контактов, временных параметрах проекта и неиспользованных ресурсах для каждого устройства в проекте. Вы можете включить в файл отчета разделы, показывающие назначения пользователя, файловую иерархию, взаимные соединения логических элементов и уравнения, реализованные в ЛЭ.

Компилятор также автоматически создает файл трассировки (**.fit**), в котором документируются назначения ресурсов и устройств для всего проекта, а также информация о трассировке. Независимо от того, успешно ли прошла трассировка, пользователь может просмотреть информацию о согласовании, разбиении и трассировки из файла согласования в окне поуровневого планировщика. Возможно также переписать назначения из файла согласования в файл назначений и конфигурации ACF для последующего редактирования.

Существует возможность дать указание модулю трассировщика (Fitter) сгенерировать выходные текстовые файлы проекта на языке AHDL (**.tdo**). Поскольку в проекте с несколькими устройствами для одного устройства генерируется один файл, следует разбить проект на несколько проектов, каждый для одного устройства, если требуется зафиксировать логику в некоторых устройствах, затем сохранить файл TDO для этого устройства как файл текстового дизайна (**.tdf**) и перекомпилировать логику для этого устройства, при этом сохраняя результаты логического синтеза, полученные в ходе

предыдущей компиляции.

Functional SNF Extractor (экстрактор для функционального тестирования)создает файл для функционального тестирования (**.snf**). Компилятор генерирует этот файл перед синтезом проекта, поэтому он содержит все узлы, присутствующие в первоначальных файлах проекта. Этот функциональный файл SNF не содержит информации о временных параметрах. Его генерация возможна только в случае, если компиляция проекта прошла без ошибок.

Timing SNF Extractor (экстрактор для тестирования временных параметров) создает (если компиляция проекта прошла без ошибок) файл для тестирования временных параметров (**.snf**), который содержит данные о временных параметрах проекта. Этот файл используется для тестирования и анализа временных параметров. Кроме того, эти файлы SNF используют также модули компилятора, содержащие программы записи в форматы EDIF, Verilog и VHDL, генерирующие выходные файлы этих форматов и также (по желанию пользователя) выходные файлы формата стандартных задержек (**.sdo**) (standart delay format output file).

Можно задать компилятору сгенерировать оптимизированный файл SNF с помощью команды Processing /Timing SNF Extractor. Оптимизация файла SNF увеличивает время компиляции, но помогает сэкономить ваше время при тестировании и анализе временных параметров.

Linked SNF Extractor (экстрактор для тестирования компоновки) создает файл (**.snf**) для тестирования компоновки при тестировании нескольких проектов (на уровне платы). Такой файл SNF комбинирует информацию из файлов SNF двух типов: для тестирования временных параметров и функционального тестирования, которые были сгенерированы для этих нескольких проектов по отдельности. Скомпонованные проекты могут использовать устройства, принадлежащие разным семействам. Если файл для тестирования компоновки содержит только информацию о временных параметрах, его можно также использовать при прогоне анализа временных параметров.

EDIF Netlist Writer (программа записи в формат EDIF). Компилятор MAX PLUS II может взаимодействовать с большинством стандартных программных средств САПР, которые могут читать файлы стандартного формата EDIF 200 или 300. Данный (необязательный) модуль компилятора, содержащий программу записи в формат EDIF, создает один или несколько выходных файлов в формате EDIF (**.edo**), содержащих информацию о функциях и (необязательно) временных параметров, полученную после проведения синтеза. Информация о временных параметрах может быть также записана в отдельные выходные файлы формата стандартной задержки (**.sdo**.

Verilog Netlist Writer (программа записи в формат Verilog)

Необязательный модуль программы записи в формат Verilog генерирует выходные файлы с расширением **.vo**, содержащие информацию о функциях и временных параметрах, полученную после проведения синтеза. Информация о временных параметров может быть также записана в отдельные выходные файлы формата стандартной задержки (**.sdo**).

VHDL Netlist Writer VHDL (программа записи в формат VHDL)

Необязательный модуль компилятора с программой записи в формат VHDL генерирует один или несколько выходных файлов (**.vho**) на языке VHDL с синтаксисом 1987 или 1993, содержащих информацию о функциях и (необязательно) временных параметров, полученную после проведения синтеза. Информация о временных параметров может быть также записана в отдельные выходные файлы формата стандартной задержки (**.sdo**).

Выходные файлы на языках описания аппаратуры можно использовать при верификации проекта с использованием внешнего симулятора. Эти файлы генерируются только в случае успешной компиляции проекта.

Assembler (модуль ассемблера)

Модуль ассемблера преобразует назначения логических элементов, выводов и устройства, сделанные модулем трассировки Fitter, в программный образ для устройства (устройств) в виде одного или нескольких двоичных объектных файлов для программатора (**.pof**) или объектных файлов SRAM (**.sof**); для некоторых устройств компилятор также генерирует ASCII-файлы JEDEC (.**jed**), содержащие информацию для программатора, конфигурационные ASCII-файлы (**.ttf**) и ASCII-файлы формата Intel (**.hex**).Объектные файлы POF и SOF, а также конфигурационные файлы JEDEC затем обрабатываются программатором системы MAX PLUS II и программирующей аппаратурой фирмы Altera (или другим программатором). Файлы HEX и TTF могут быть использованы для конфигурирования устройств FLEX 6000, FLEX 8000 и FLEX 10K с помощью других средств. Модуль ассемблера создает файлы для программатора только в случае успешной компиляции проекта.

После завершения компиляции компилятор и программатор системы MAX PLUS II позволяют сгенерировать дополнительные файлы для программирования устройства, которые можно использовать в других условиях программирования. Например, можно создать файлы с последовательным потоком битов (**.bbs**) и необработанные двоичные файлы (**.rbf**) для конфигурирования устройств FLEX 6000, FLEX 8000 и FLEX 10K. Можно создать последовательные файлы тестовых векторов (**.svf**) или файлы на языке JAM (**.jam**) для программирования устройств в автоматизированной испытательной аппаратуре типа ATE.

Design Doctor Utility (утилита диагностики проекта). Необязательная утилита диагностики проекта проверяет логику каждого файла проекта для выявления элементов, которые могут вызвать проблемы надежности на системном уровне. Эти проблемы обычно обнаруживаются только после запуска устройства “в железе”. Существует возможность выбора одного из трех предварительно определенных наборов правил разработки проекта с разными уровнями. Кроме того, вы можете разработать свой собственный свод правил разработки дизайна.

Правила разработки дизайна основываются на принципах надежности, которые охватывают логику, содержащую асинхронные входы, залповые тактовые сигналы (Clock), многоуровневую логику на конфигурациях с сигналами Clock, Preset и Clear, а также в условиях состязаний. Установка правил проверки производиться с помощью команды Design Doctor Settings (рис.2.27)

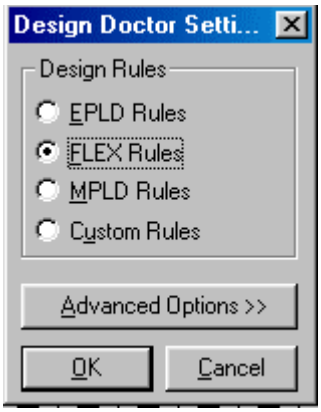

Рис.2.27. Окно команды Design Doctor Settings

2.5. Верификация проекта

Для верификации проекта (см. рис. 2.28) в системе MAX PLUS II используются три приложения: симулятор (Simulator), анализатор временных параметров (Timing Analyzer) и сигнальный редактор (Waveform Editor).

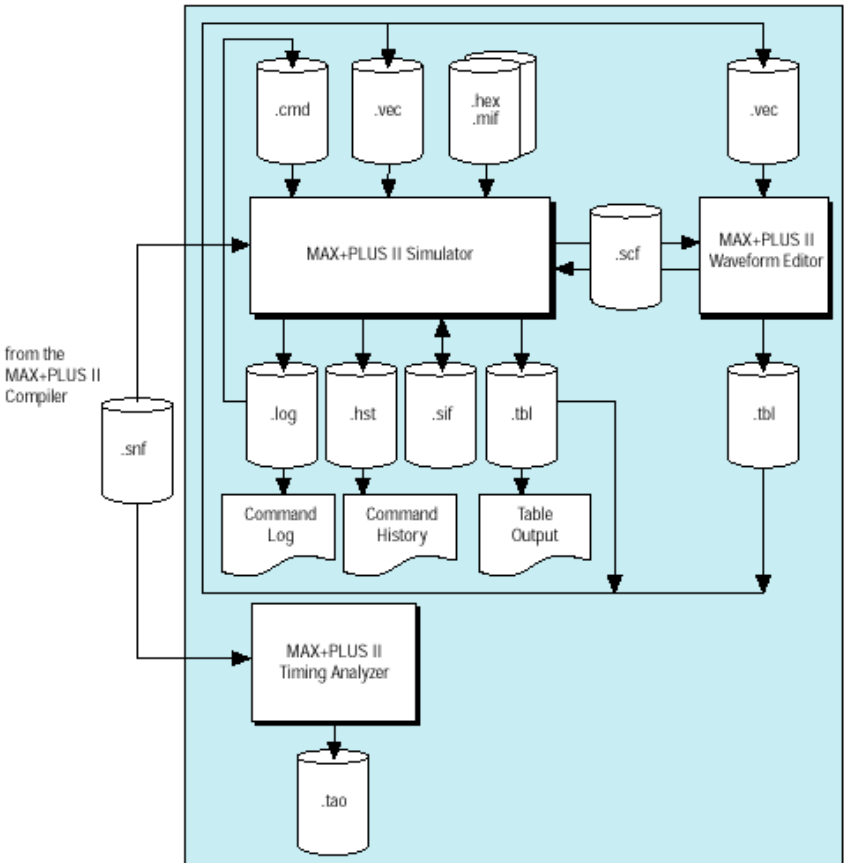

Рис. 2.28. Верификация проекта в системе MAX PLUS II

Симулятор(Simulator)

Симулятор системы MAX PLUS II тестирует логические операции и внутреннюю синхронизацию проекта, позволяя пользователю моделировать проект. Симулятор может работать как в диалоговом или автоматическом (пакетном) режимах. Окно симулятора показано на рис. 2.29.

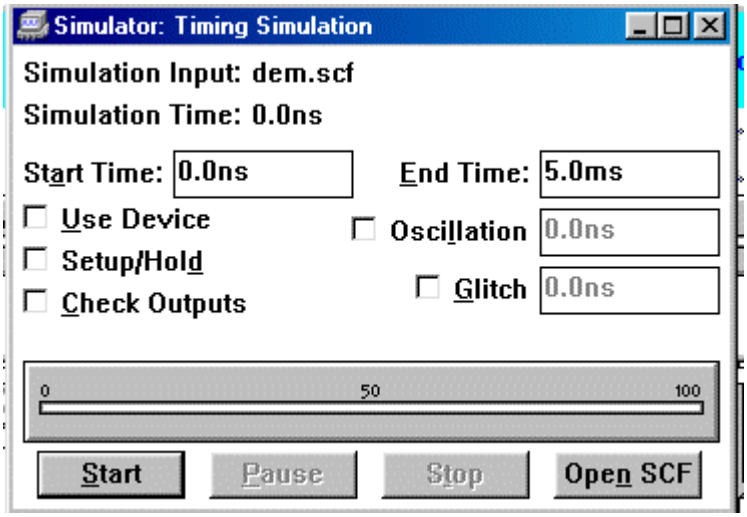

Рис. 2.29. Симулятор MAX PLUS II

Перед тестированием проект необходимо скомпилировать, задав компилятору опцию сгенерировать файл (**.snf**) для функционального тестирования, тестирования временных параметров, либо тестирования компоновки нескольких проектов

(устройств). Затем полученный для текущего проекта файл SNF загружается автоматически при открытии симулятора.

В качестве источника входных векторов используется либо графический сигнальный файл каналов тестирования (**.scf**), либо текстовый ASCII-файл (**.vec**). Для проектов, работающих с памятью, можно задать некое исходное содержимое памяти в файлах шестнадцатеричного формата (Intel) с расширением **.hex** или в файлах инициализации памяти с расширением **.mif**. Сигнальный редактор может автоматически создавать файл SCF по умолчанию, который пользователь может редактировать с целью получения нужных ему тестовых входных векторов. Если вместо этого используется текстовый ASCII-файл векторов, сигнальный редактор автоматически генерирует из него файл каналов тестирования SCF(simulator chanel file).

Симулятор позволяет проверить выходные значения, получаемые в ходе тестирования, в по выходам, содержащимся в файле SCF (заданным пользователем прогнозируемым значениям или результатам предыдущих тестов). С помощью соответствующей аппаратуры для программирования можно также выполнить функциональное тестирование для проверки действительных значений выходов программируемого устройства по результатам тестирования.

Используя различные опции симулятора можно контролировать проект на появление сбоев (glitches) , а также нарушение установочных параметров и временных задержек. После завершения тестирования можно открыть сигнальный редактор для просмотра обновленного файла SCF или сохранить полученные выходные значения в табличном файле с расширением **.tbl**, а затем просматривать результаты в текстовом редакторе.

Функциональное тестирование. Если компилятору “дано задание” сгенерировать файл SNF для функционального тестирования, он создает его перед синтезом проекта. Следовательно, при функциональном тестировании можно смоделировать все узла проекта.

Во время функционального тестирования симулятор игнорирует все задержки распространения. Поэтому в файле SNF для функционального тестирования нет задержек, выходные логические уровни изменяются одновременно со входными векторами.

Тестирование временных параметров

Файл SNF для тестирования временных параметров компилятор генерирует после того как проведены полный синтез и оптимизация проекта. Поэтому этот файл содержит только те узлы, которые не были уничтожены в процессе логического синтеза.

Из этого файла симулятор берет информацию об аппаратной части, которая была собрана из файлов моделей устройств (**.dmf**), имеющихся в комплекте системы MAX PLUS II.

Если проект был разбит на несколько устройств, компилятор создает файл SNF для проекта в целом и для каждого устройства. Однако тестирование временных параметров осуществляется только для проекта в целом.

Можно ускорить тестирование временных параметров, выдав компилятору указание сгенерировать оптимизированный файл SNF, содержащий динамические модели, которые представляют собой разные типы комбинаторной логики. Процессорное время компилятора увеличивается при этом, однако полученный оптимизированный SNF может уменьшить время тестирования, поскольку симулятор может работать с динамическими моделями вместо того, чтобы интерпретировать всю логику в комбинаторной схеме.

При создании файла SNF для тестирования компоновки нескольких проектов компилятор комбинирует файлы SNF для функционального тестирования и/или файлы для тестирования временных параметров нескольких отдельных проектов. Отдельные “подпроекты” в компоновочном SNF могут быть предназначены для устройств разных семейств. Кроме того, поскольку файлы SNF для функционального тестирования создаются до окончания полной компиляции, можно ввести подпроекты, которые представляют логику, не реализованную в устройстве фирмы Altera.

Компоновочный файл SNF можно использовать для тестирования на уровне платы. Кроме того, если он содержит только информацию о временных параметрах, его можно использовать для прогона временного анализатора системы MAX PLUS II.

Вместе с другими приложениями системы MAX PLUS II симулятор позволяет вам выполнить следующие задачи:

- задать ожидаемые логические уровни на выходе, которые можно будет сравнить с результатами тестирования;

- смоделировать отдельные узлы или узлы, объединенные в группы. Можно комбинировать биты цифрового автомата в

проекте, моделировать их как группу и обращаться к ним по имени состояния;

- определить временной интервал, представляющий собой дрожание фронта импульса или сбой, и проанализировать проект на наличие обоих этих условий или одного из них;

- контролировать наличие в проекте нарушений начальных установок регистров и временных задержек;

- регистрировать действительные значения выходов устройств вместо моделированных;

- проводить функциональное тестирование. Можно проверить, являются ли смоделированные выходные значения функционально эквивалентными реальным выходам устройств;

- задавать условия точки останова, которые заставляют симулятора делать паузу при их реализации в процессе тестирования;

- составлять перечень имен и логических уровней любой комбинации узлов и групп и инициализировать логические уровни узла или группы перед тестированием;

- инициализировать содержимое блоков памяти RAM (ОЗУ) или ROM (ПЗУ) перед тестированием;

- сохранять инициализированные значения узлов и групп, в том числе инициализированное содержимое памяти, в файле инициализации симулятора (**.sif**) или перезагружать инициализированные значения , хранящиеся в файле;

- регистрировать команды симулятора в текстовом файле протокола испытаний (**.log**) того же формата, что и командный файл (**.cmd**), используемый при тестировании в автоматическом (пакетном) режиме, а затем использовать этот файл LOG для повторения этого цикла тестирования. Команды симулятора и полученные результаты можно также записать в тестовый файл истории тестирования с расширением **.hst**.

Таким образом, мы рассмотрели основные приемы работы пакета MAX+PLUS II. Конечно в рамках одной главы практически невозможно подробно рассмотреть все приемы работы с таким сложным и разнообразным программным средством, однако заинтересованный пользователь в состоянии самостоятельно освоить пакет, используя данную книгу и фирменное руководство пользователя .