

## Использование САПР MAX+PLUS II для разработки цифровых устройств на ПЛИС фирмы «Альтера»

САПР «MAX+PLUS II» представляет собой интегрированную среду для разработки цифровых устройств на базе программируемых логических интегральных схем (ПЛИС) фирмы «Altera» («Альтера») и обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

- создание проектов устройств;
- синтез структур и трассировку внутренних связей ПЛИС;
- подготовку данных для программирования или конфигурирования ПЛИС (компиляцию);
- верификацию проектов (функциональное моделирование и временной анализ);
- программирование или конфигурирование ПЛИС.

### Приложения системы «MAX+PLUS II»

В состав пакета «MAX+PLUS II» входят следующие связанные между собой приложения, реализующие все перечисленные выше этапы разработки цифровых устройств на ПЛИС фирмы «Альтера»:

#### Приложения для ввода проектов (редакторы проектов)

«**Graphic Editor**» – графический редактор, предназначенный для ввода проекта в виде схемы соединений символов элементов, извлекаемых из стандартных библиотек пакета либо из библиотеки пользователя.

«**Waveform Editor**» – редактор временных диаграмм (сигнальный редактор), который выполняет двойную функцию: на этапе ввода обеспечивает ввод логики проекта в виде диаграмм (эпюр) состояний входов и выходов, а на этапе моделирования обеспечивает ввод диаграмм тестовых (эталонных) входных состояний моделируемого устройства и задание перечня тестируемых выходов.

«**Text Editor**» – текстовый редактор, предназначенный для создания и редактирования текстовых файлов, содержащих описание логики проекта на языке описания устройств AHDL (Altera Hardware Description Language) или на близких к нему языках типа VHDL, "Verilog".

«**Symbol Editor**» – символьный редактор, позволяющий редактировать существующие символы и создавать новые. Кстати, любой откомпилированный проект может быть свёрнут в символ, помещён в библиотеку символов и использован как элемент в любом другом проекте.

«**Floorplan Editor**» – редактор связей (поуровневый планировщик), который на плане расположения основных логических элементов позволяет вручную распределять выводы ПЛИС (закреплять выводы за конкретными входными и выходными сигналами) и перераспределять некоторые внутренние ресурсы ПЛИС.

#### Приложения «MAX+PLUS II Compiler»

Это приложения, входящие в пакет компилятора и предназначенные для синтеза структуры, трассировки связей, проверки корректности проекта и локализации ошибок, формирования файлов программирования или конфигурирования ПЛИС:

«**Compiler Netlist Extractor**» – приложение, обеспечивающее извлечение списка соединений из исходного файла представления проекта, созданного при вводе проекта.

«**Database Builder**» – приложение, предназначенное для построения базы данных проекта.

«**Logic Synthesizer**» – приложение, обеспечивающее проверку корректности проекта по формальным правилам и синтез оптимальной структуры проекта.

«**Partitioner**» – приложение, обеспечивающее разбиение проекта на части в тех случаях, когда ресурсов одного кристалла (микросхемы) недостаточно для реализации проекта.

«**Fitter**» – трассировщик внутренних связей, обеспечивающий реализацию синтезированной структуры.

«**Timing SNF Extractor**» – приложение, обеспечивающее извлечение параметров проекта, необходимых для функционального моделирования и временного анализа.

«**Assembler**» – приложение, создающее из скомпилированного проекта файлы, необходимые для программирования или конфигурирования ПЛИС через программатор.

### Приложения для верификации проектов

«**Simulator**» – приложение, которое совместно с редактором временных диаграмм предназначено для функционального моделирования проекта с целью проверки правильности логики его функционирования.

«**Timing Analyzer**» – приложение, обеспечивающее расчет временных задержек от каждого входа до каждого логически связанного с ним выхода.

### «MAX+PLUS II Programmer»

Наконец, для программирования или конфигурирования ПЛИС используется приложение «**MAX+PLUS II Programmer**».

Программирование и перепрограммирование микросхем, имеющих встроенную систему программирования (ISP), может осуществляться непосредственно в составе конечного изделия через специальный кабель, подключаемый либо к LPT-порту (Byte Blaster), либо к COM-порту (Bit Blaster) компьютера и технологического 10-контактного соединителя интерфейса JTAG, устанавливаемого на плате изделия.

Схемы кабелей можно найти на сайте фирмы «Альтера» в разделе «Development Tools». Если на плате изделия устанавливается несколько ПЛИС со встроенными системами программирования, то все они могут программироваться через один технологический разъём. Для этой цели приложение «Programmer» имеет режим «Multi-Device» (к сожалению, бесплатные версии пакета этот режим не поддерживают). Схемы подключения ПЛИС к интерфейсу JTAG приводятся в документации на «Bit Blaster» и «Byte Blaster».

Для программирования остальных микросхем необходимо дополнительно использовать внешний программатор, который также может подключаться к COM- или LPT-порту.

### Сервисные приложения

В состав САПР, кроме того, входят три сервисных приложения:

«**Design Doctor**» – приложение, предназначенное для проверки корректности проекта с использованием эмпирических правил.

«**Message Processor**» – процессор сообщений, обеспечивающий обработку, вывод на отображение и локализацию (указание места в проекте, к которому оно относится) сообщений трёх типов: сообщений об ошибках («Error»), предупреждений («Warning») и информационных сообщений («Info»). Причину вывода того или иного сообщения можно выяснить через опцию «Help on Message» процессора сообщений.

При наличии сообщений об ошибках компиляция проекта невозможна до их полного устранения. При наличии предупреждений компиляция успешно завершается, однако наличие предупреждения свидетельствует об обнаружении проблемы, которая может привести к неверной работе устройства. Поэтому все предупреждения должны быть тщательно проанализированы с использованием «Help on Message», до выяснения причин их появления и последующего устранения этих причин (или игнорирования предупреждения, что иногда бывает возможно). Информационные сообщения нужно только принимать к сведению.

«**Hierarchy Display**» – приложение, обеспечивающее обзор иерархической структуры проекта, который может состоять из множества составленных в различных редакторах и свёрнутых в символы проектов более низких уровней, причём число уровней не ограничивается. Основной проект (проект самого верхнего уровня) должен быть создан в графическом редакторе (если проект имеет только один уровень иерархии, то он может быть создан в любом редакторе).

## Рабочие каталоги системы

Во время инсталляции пакета создаются два каталога: каталог \MAXPLUS2, который содержит все приложения и библиотеки пакета, и каталог \MAX2WORK, который содержит подкаталог \CHIPTRIP со всеми файлами учебного проекта, рассмотренного в руководстве («MAX+PLUS II Tutorial»), и ряд подкаталогов, используемых электронным справочником («MAX+PLUS II Help»). В этом же каталоге \MAX2WORK следует размещать и рабочие каталоги создаваемых проектов устройств.

Необходимость создания отдельных каталогов для каждого разрабатываемого проекта обусловлена тем, что в процессе разработки проекта системой MAX+PLUS II создаётся и поддерживается множество файлов, относящихся к текущему проекту. Прежде всего это файл проекта («Project File»), название которого определяет название проекта в целом. Этот файл содержит основную логику и иерархию проекта, обрабатываемую компилятором.

Кроме того, создаётся ряд вспомогательных файлов, связанных с проектом, но не являющихся частью логики проекта. Большая часть вспомогательных файлов создаётся и автоматически помещается в каталог проекта в процессе ввода и компиляции проекта. Это прежде всего файлы назначений и конфигурации (.ACF), файлы отчётов (.RPT), файлы данных для функционального моделирования и временного анализа (.SNF), файлы данных для программирования (.POF) и ряд других. Названия этих файлов всегда совпадают с названием проекта.

Некоторые вспомогательные файлы создаются пользователем: например, для выполнения функционального моделирования создаётся файл (.SCF), содержащий описание начальных и текущих состояний входных сигналов (входов) и перечень выходов, для которых должны быть определены выходные сигналы. Поэтому перед началом работы над новым проектом следует создать рабочий каталог проекта, при этом имя каталога можно выбирать произвольно (т.е. имя каталога может не совпадать с именем файла проекта).

## Начало работы

После запуска пакета открывается главное окно. В самой верхней строке содержится имя последнего проекта, с которым велась работа. Две следующие строки являются обычными для Windows-приложений: строка основного меню и панель инструментов, в левой части которой расположены стандартные функции Windows («New», «Open», «Save», «Print», «Cut», «Copy», «Paste», «Undo»), а в правой части – специфические инструменты пакета, с помощью которых осуществляется запуск его основных приложений.

Назначение любого инструмента можно узнать, совместив с ним указатель мыши, при этом в строке подсказки (самая нижняя строка окна) появляется текст с пояснением. Для запуска приложений можно использовать также опции меню «MAX+PLUS II» (в основном меню).

Для начала, используя стандартные средства Windows, создадим рабочий каталог под именем \MOD\_IND.

Затем создадим проект схемы формирования сигнала записи в порт. Будем создавать его в графическом редакторе. Для этого запускаем систему «MAX+PLUS II» и создаём новый файл (крайний левый значок панели инструментов). В открывшемся диалоговом окне «New» выбираем опцию «Graphic Editor file» и нажимаем кнопку «OK», при этом автоматически открывается окно графического редактора.

Окно редактора имеет ряд дополнительных пунктов основного меню и панель инструментов редактора, расположенную вертикально с левой стороны окна. Сохраним новый файл проекта (через меню «File > Save») под именем WRITE\_PORT (расширение будет присвоено автоматически) в созданном каталоге \MOD\_IND. Имя файла проекта следует обязательно привязать к имени проекта – это делается при выборе параметра «Set Project to Current File» (в опции «Project» меню «File» главного меню рабочего окна).

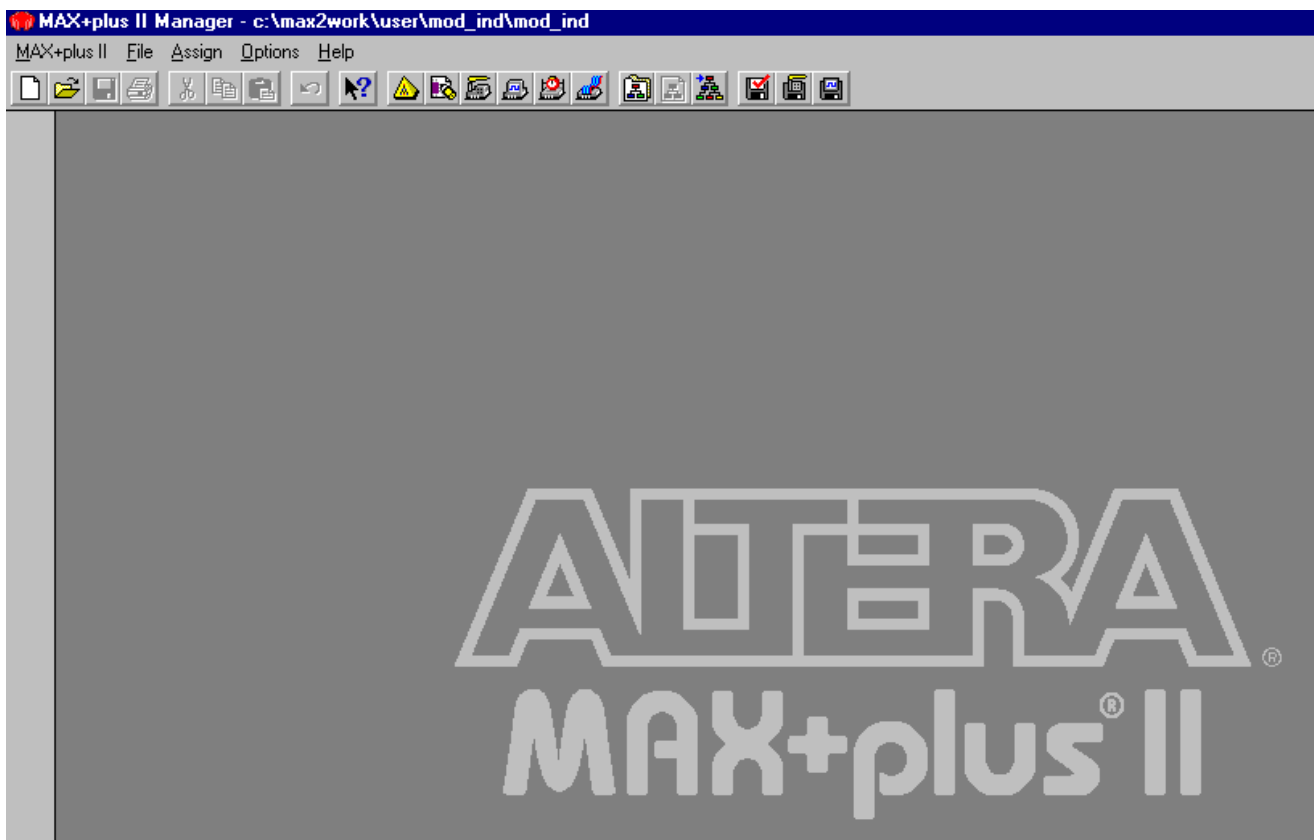


Рис. 1. Основная часть главного окна системы «MAX+PLUS II»

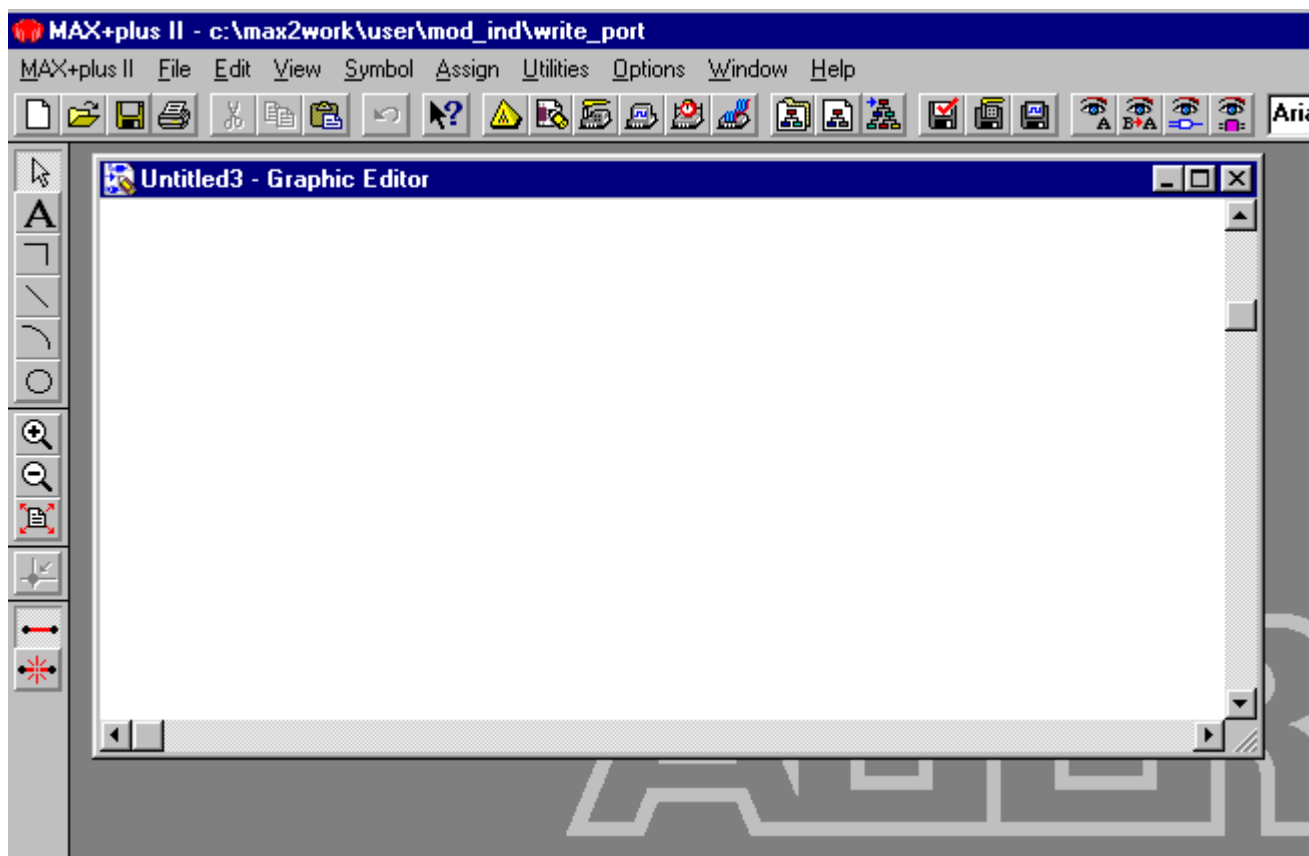


Рис. 2. Окно графического редактора «Graphic Editor»

Для создания графического проекта можно использовать библиотеки примитивов (\MAXPLUS\MAX2LIB\PRIM), макрофункций (\MAXPLUS\MAX2LIB\MF) и параметризованных мегафункций (\MAXPLUS\MAX2LIB\MEGA\_LPM).

Примитивы включают большой набор основных логических элементов, триггеров, элементов входа и выхода (INPUT, OUTPUT, BIDIR), а также вспомогательные элементы: GND (логический ноль), VCC (логическая единица).

Макрофункции включают главным образом эквиваленты логических микросхем 74-й серии.

Параметризованные мегафункции позволяют реализовывать многовходовые и много-разрядные элементы цифровой схемотехники (логику, регистры, мультиплексоры и т.д.), вводя ряд параметров в специально выделенных областях условных графических обозначений этих элементов. Порядок использования мегафункций можно найти в соответствующем разделе электронного справочника системы («Help»), а также в документе LPM Quick Reference Guide на сайте фирмы «Альтера».

### Работа в графическом редакторе

Прежде чем перейти непосредственно к созданию проекта схемы формирования сигнала записи в порт, рассмотрим некоторые особенности и свойства графического редактора системы «MAX+PLUS II».

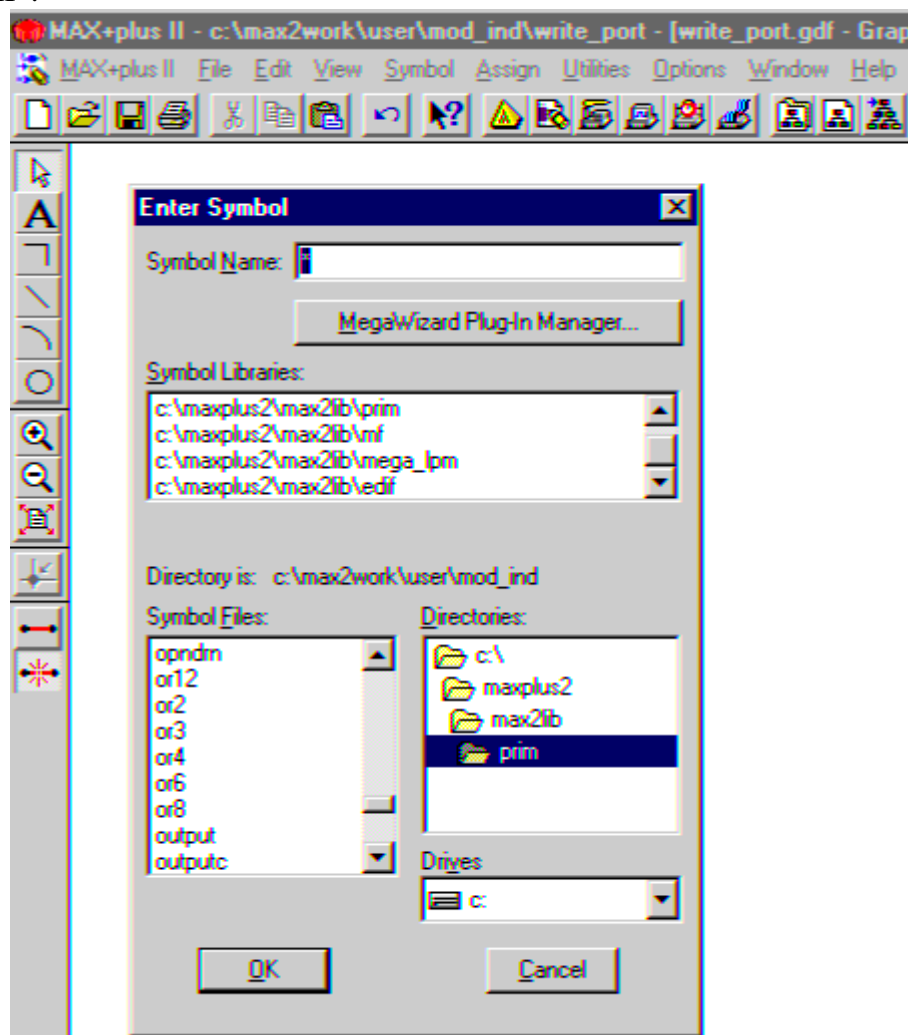


Рис. 3. Диалоговое окно размещения элементов

Для размещения элементов в окне редактора используется диалоговое окно «Enter Symbol», которое можно открыть через меню «Symbol» основного меню редактора (опция «Enter Symbol»). Однако удобнее вызывать это окно двойным щелчком левой клавиши мыши по свободному пространству открытого окна редактора, после чего выбранный элемент будет размещён именно в этом месте (позиционирование элемента производится по верхнему левому углу условного обозначения элемента). Выбрать и установить необходимый элемент можно двумя способами:

1. Набрать имя элемента (примитива, мега- или макрофункции) в окне «Symbol Name» диалогового окна «Enter Symbol» и нажать «ОК».
2. Выбрать необходимую библиотеку в окне «Symbol Libraries» диалогового окна «Enter Symbol» и двойным нажатием левой клавиши мыши открыть её. Затем аналогичным образом выбрать необходимый элемент в окне «Symbol File».

Установленный элемент можно переместить в другое место обычным приёмом «Drag&Drop». Перемещение элементов можно производить без сохранения или с сохранением соединений, проведённых между элементами, в зависимости от состояния инструмента «Rubberbanding Functions» (ON – связи сохраняются, OFF – не сохраняются). Кнопки управления этим инструментом расположены в нижней части панели инструментов редактора.

Выводы символов элементов можно соединить сигнальными линиями или присваиваемыми одинаковыми именами. Для проведения сигнальной линии необходимо совместить указатель курсора (конец стрелки) с выводом элемента, при этом курсор автоматически превращается в инструмент рисования ортогональных линий (перекрестие). После этого проводится необходимая линия (при нажатой и удерживаемой левой кнопке мыши). За один приём можно провести два ортогональных отрезка. Если этого недостаточно, то процедуру можно повторять, начиная с конца проведённой линии или с вывода другого элемента.

Для соединения элементов необязательно проводить сплошные соединительные линии между выводами. Достаточно выводам, которые должны быть соединены между собой, присвоить одинаковые имена. Поименованный вывод элемента называется узлом («Node»). Для присвоения узлу имени необходимо сначала провести короткую сигнальную линию. Так как по окончании рисования линии она остаётся активной (активные или выделенные элементы обозначаются красным цветом или красной рамкой), имя узла можно вводить сразу. Если линия деактивирована (обозначена чёрным цветом), то необходимо подвести курсор к нужному месту начала имени узла на линии, щелчком левой кнопки мыши активировать её и ввести требуемое имя с клавиатуры.

Установленные элементы и группы элементов (любую выделенную область окна) можно удалять, перемещать, поворачивать, копировать и размножать в пределах окна редактора, а также переносить в другие окна через буфер обмена. Все эти операции можно выполнять с использованием опций меню «Edit» (в основном меню редактора), предварительно выделив требуемую область или элемент.

Выделение отдельных элементов, включая и соединительные линии, производится щелчком левой кнопки мыши на изображении символа элемента. Выделение области возможно при нажатой и удерживаемой левой кнопке мыши. Выделенная область обозначается рамкой чёрного цвета, а все выделенные элементы – красным цветом.

Для удаления, копирования, вырезания, поворота и некоторых других операций с элементами можно использовать выпадающее контекстное меню, которое вызывается щелчком правой кнопки мыши по выбранному элементу. Для перемещения выделенных областей в пределах окна используется опция «Drag&Drop», а для размножения выделенных областей и элементов – «Drag&Drop» при нажатой и удерживаемой клавише CTRL.

Часть инструментов редактора, расположенных на вертикальной панели с левой стороны окна, предназначена для создания поясняющих надписей, рисунков и таблиц, которые не являются частью проекта и не обрабатываются компилятором.

### **Этап 1: создание схемы проекта**

Создание схемы проекта целесообразно начинать с размещения узлов входов и выходов и присвоения этим узлам имён. В нашем случае необходимо разместить входы адресных линий A0...A15, вход сигнала записи /WR (низкий активный уровень) и один выход сигнала записи в порт WR\_PORT. Используем для этой цели библиотеку примитивов, которая содержит примитивы INPUT (для входов) и OUTPUT (для выходов). Каждому вводу должно быть присвоено своё имя. Для этого необходимо поместить указатель мыши в поле имени ввода и дважды

нажать левую кнопку мыши. При этом появляется чёрный прямоугольник, который сигнализирует о возможности набора нового имени. Переход к набору имени следующего элемента ввода может осуществляться по нажатию клавиши ENTER.

Логика разрабатываемого проекта также может быть реализована с использованием примитивов: двух элементов OR8 (8-входовый элемент «ИЛИ»), одного элемента NOR3 (3-входовый элемент «ИЛИ-НЕ») и одного инвертора NOT. В описанном выше порядке размещаем требуемые элементы и выполняем необходимые соединения.

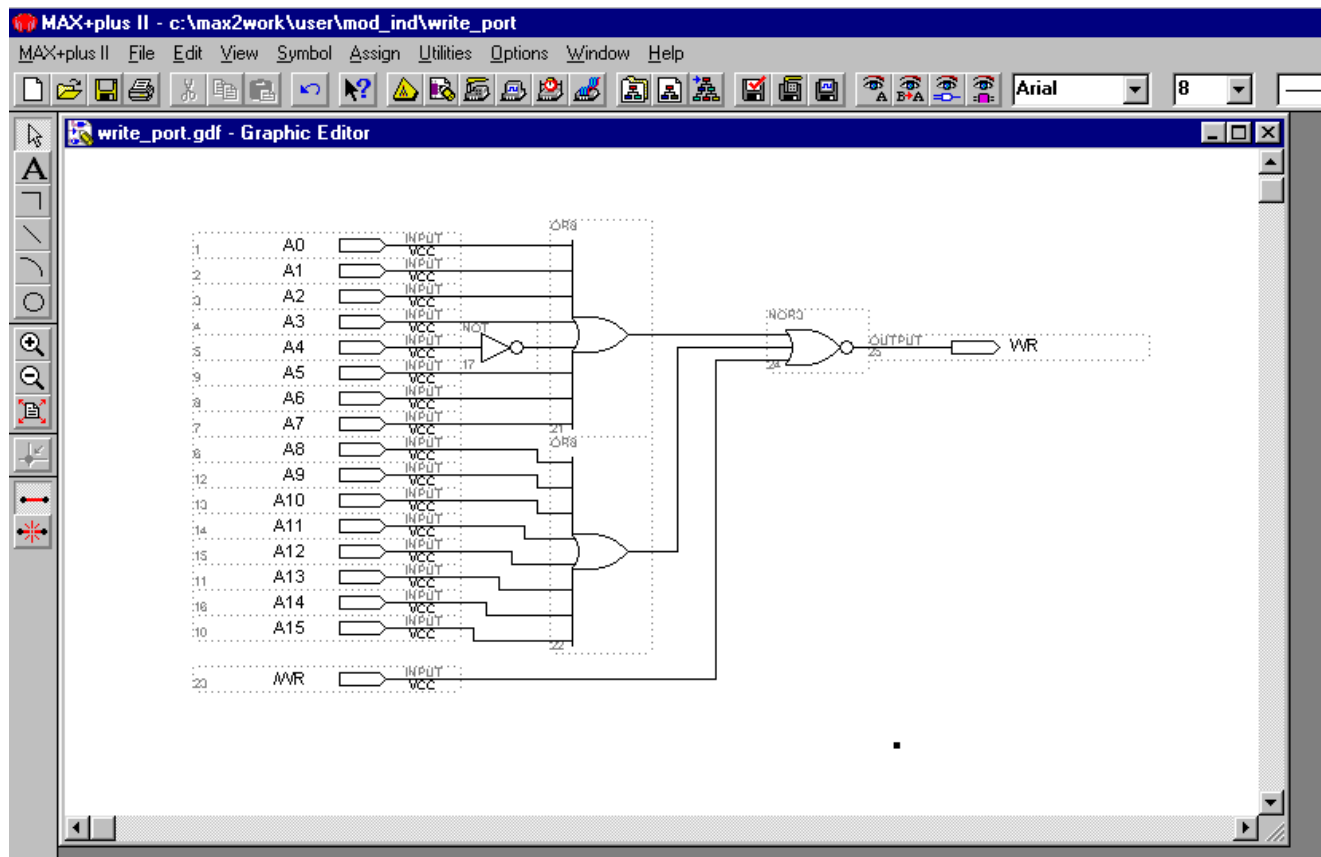


Рис. 4. Готовый проект устройства формирования сигнала записи в порт

Заметим, что предложенный вариант реализации логики устройства – не единственный и не самый лучший. Заинтересованный читатель может реализовать этот проект с использованием языка AHDL (самый предпочтительный вариант) или мегафункции LPM\_OR.

## Этап 2: компиляция и создание символа проекта

Следующий этап – компиляция и создание символа проекта для включения его в файл проекта верхнего уровня. Перед компиляцией можно выполнить проверку корректности введённого проекта. Проверка осуществляется через опцию «Project» (меню «File» главного меню рабочего окна) путём выбора параметра «Save&Check» или щелчком левой кнопки мыши на пиктограмме соответствующего инструмента на основной панели.

Компиляция осуществляется через опцию «Project» после выбора параметра «Save&Compile» или опять же с помощью соответствующего инструмента с основной панели. По результатам компиляции процессор сообщений проинформирует нас о том, что для реализации данного проекта могут быть использованы ПЛИС семейства «MAX3000A», а именно ПЛИС типа «EPM 3032ALC44» (в корпусе PLCC с 44 выводами).

Создание символа проекта осуществляется через опцию «Project», в котором следует выбрать параметр «Create Default Symbol» – этот параметр становится доступным только после закрытия окна компилятора. Созданный символ будет помещён в каталог проекта. Использование созданных символов, так же как и элементов других библиотек, производится через диалоговое окно «Enter Symbol».

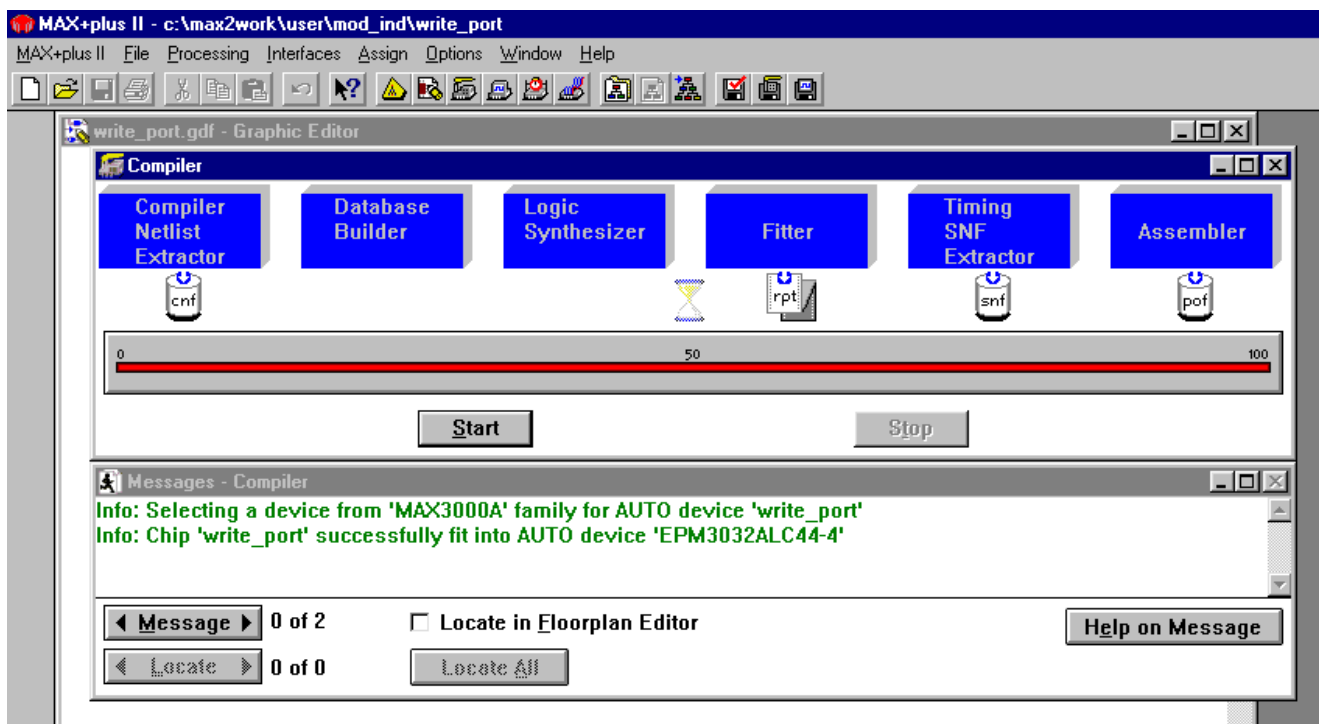


Рис. 5. Вид окна компилятора после успешного завершения компиляции

### Этап 3

Следующий шаг – создание проекта и символа комбинационной схемы преобразования двоичного кода тетрады в сигналы управления сегментами индикатора.

Для создания проекта этого устройства используем редактор временных диаграмм («Waveform Editor»). Для этого следует нажать кнопку открытия нового файла на панели инструментов, в открывшемся диалоговом окне «New» отметить опцию «Waveform Editor file», в соседнем окне выбрать расширение .WDF (расширение .SCF используется для моделирования) и нажать «OK», после чего открывается окно редактора.

Окно редактора имеет четыре поля, разделённых вертикальными линиями. Первое поле слева («Name») предназначено для ввода имени узла, во втором поле («Type») отображается тип ввода (INPUT, OUTPUT, BIDIR), в третьем поле («Value») показаны состояния выводов, соответствующие положению специальной вертикальной визирной линии, которая при открытии окна установлена в начало горизонтальной оси, размеченной в единицах времени.

Для перемещения визира в нужное место необходимо совместить курсор с вершиной визира (при этом курсор примет крестообразную форму) и при нажатой левой кнопке мыши перетащить визир в нужное место.

Четвёртое поле предназначено для задания требуемых состояний выводов, при этом используются инструменты с панели инструментов редактора, которая расположена вертикально вдоль левой стороны окна. Активизация панели инструментов происходит только если выделен один из узлов. Чтобы выделить узел, необходимо совместить курсор с именем узла и щёлкнуть левой кнопкой мыши, можно также выделить любой участок вдоль горизонтальной оси, при этом границы выделяемых участков привязываются к сетке.

Параметры сетки устанавливаются следующим образом: с помощью опции «End Time» (меню «File») задаётся максимальное значение временного интервала с указанием единиц измерения, а с помощью опции «Grid Size» – шаг сетки. В верхней части экрана расположены окна для точного отсчёта интервалов времени.

Сохраним созданный файл в каталоге \MOD\_IND под именем CONV\_BIN\_HEX и присвоим проекту имя созданного файла (через опцию «Project» меню «File» главного меню рабочего окна, путём выбора параметра «Set Project to Current File»).

Для ввода имени узла можно использовать опцию «Insert Mode» меню «Node».



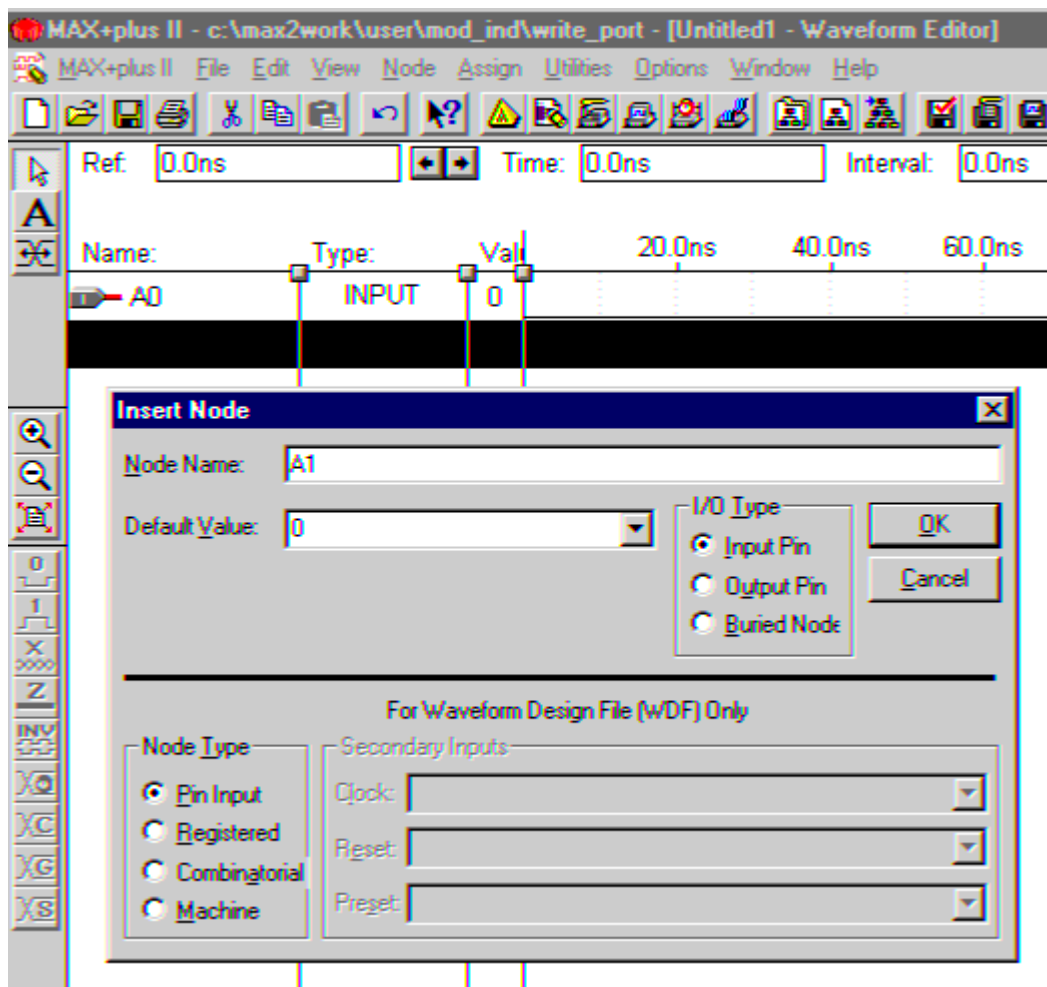


Рис. 6. Диалоговое окно опции «Insert Mode»

В верхнее поле «Node Name» необходимо ввести имя узла, в окне «I/O Type» указать тип вывода и нажать «OK». Кроме того, имена узлов можно вводить другим способом. Для этого необходимо поместить курсор в поле «Name» и щёлкнуть левой кнопкой мыши, при этом автоматически открывается диалоговое окно «Insert Mode».

Размещённые узлы можно редактировать, перемещать, удалять, размножать (с обязательным редактированием имени или типа, если это необходимо). Для редактирования используется то же диалоговое окно «Insert Mode». Остальные операции осуществляются аналогично тому, как это делается в графическом редакторе.

В нашем случае необходимо задать 16 возможных состояний 4-х входов и состояния 7-ми выходов, соответствующие каждому состоянию входов. Для этого вся горизонтальная ось должна быть разбита на 16 дискрет. Например, установим значение «End Time» 160 нс, а «Grid Size» – 10 нс. После этого последовательно выделяем узлы и дискреты и с помощью панели инструментов задаём требуемые состояния входов и выходов.

Далее компилируем файл и создаём символ этого устройства, как описано выше.

После этого в графическом редакторе создаем и компилируем файл проекта всего устройства в целом. В качестве регистра порта используем макрофункцию 74374, а в качестве остальных элементов – элемент WRITE\_PORT и два элемента CONV\_BIN\_HEX.

Порядок открытия, создания и компиляции файла проекта верхнего уровня ничем не отличается от порядка создания и компиляции файла WRITE\_PORT, который был рассмотрен выше.

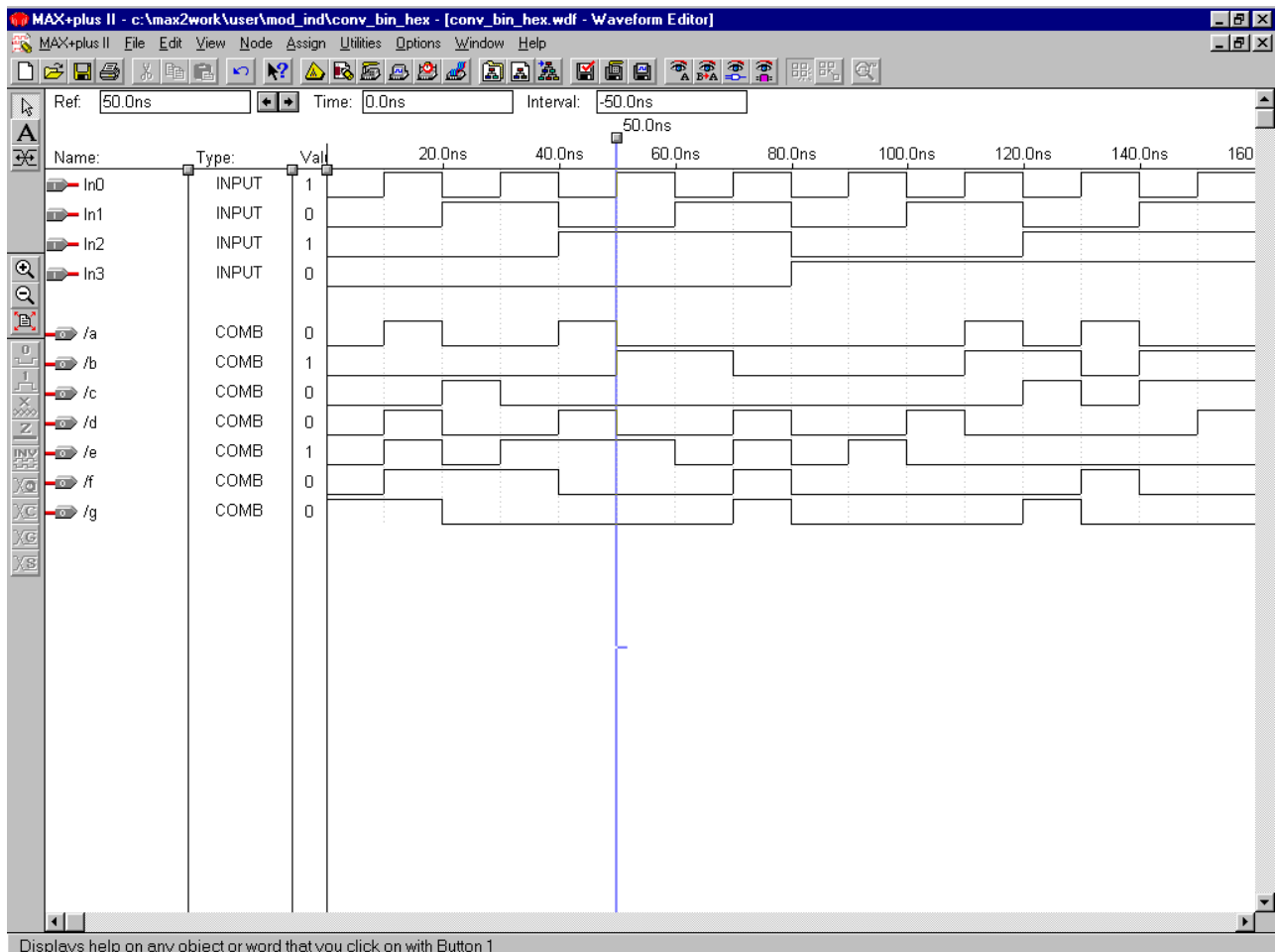


Рис. 7. Введённый файл проекта примера

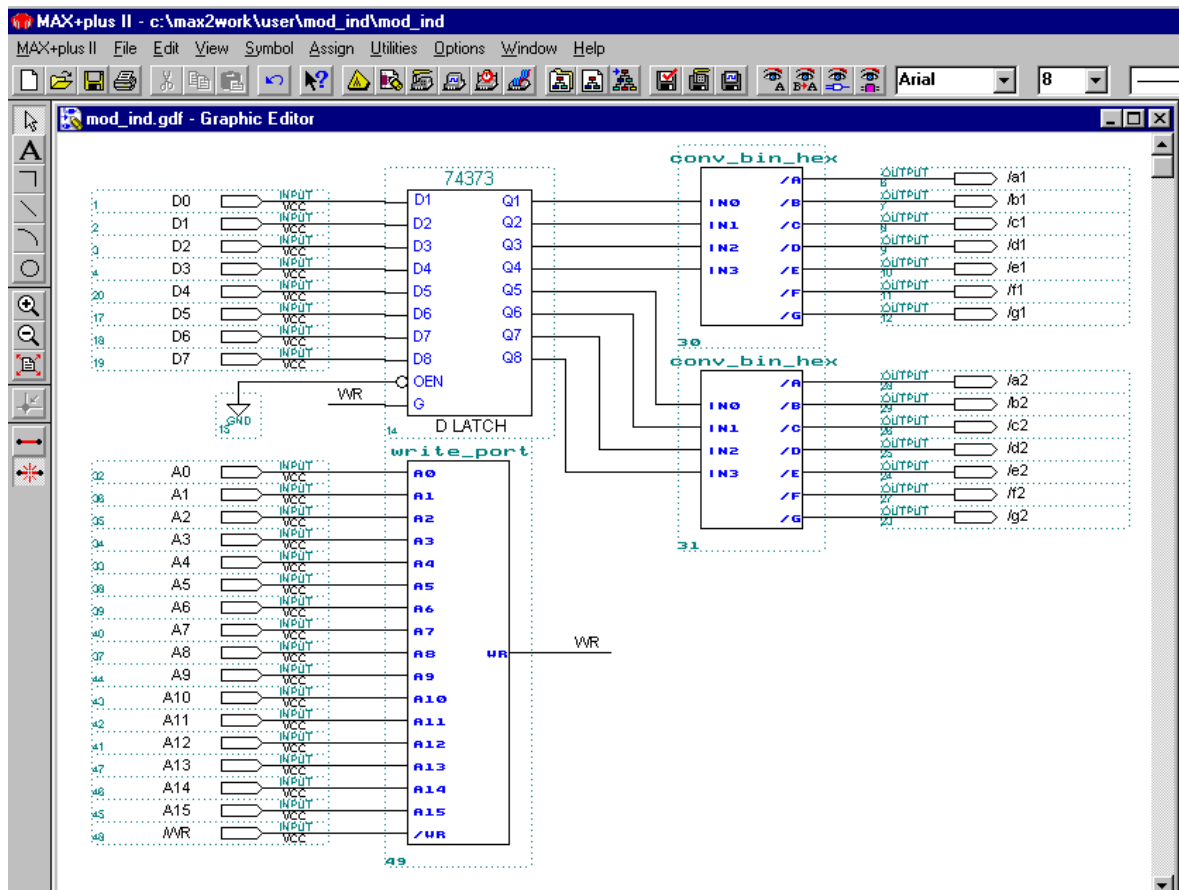


Рис. 8. Схема описанного в примере проекта устройства индикации

## Назначение типа ПЛИС и выводов ПЛИС

Тип ПЛИС, необходимый для реализации проекта, может быть выбран автоматически или назначен вручную. При создании файла проекта по умолчанию установлен режим автоматического выбора минимальной по объёму ПЛИС, в которой может быть реализован данный проект. При необходимости тип ПЛИС может быть назначен вручную с помощью опции «Device» (меню «Assign» основного меню системы). После назначения устройства проект необходимо перекомпилировать.

Выводы ПЛИС также первоначально назначаются автоматически. После завершения работы над проектом необходимо закрепить или переназначить выводы ПЛИС, для того чтобы при возможной последующей доработке (отладке) проекта в составе всего изделия компилятор не мог изменить их назначение. Эта операция выполняется с помощью редактора «Floorplan Editor», который запускается либо через меню «MAX+PLUS II» (в основном меню), либо через панель инструментов.

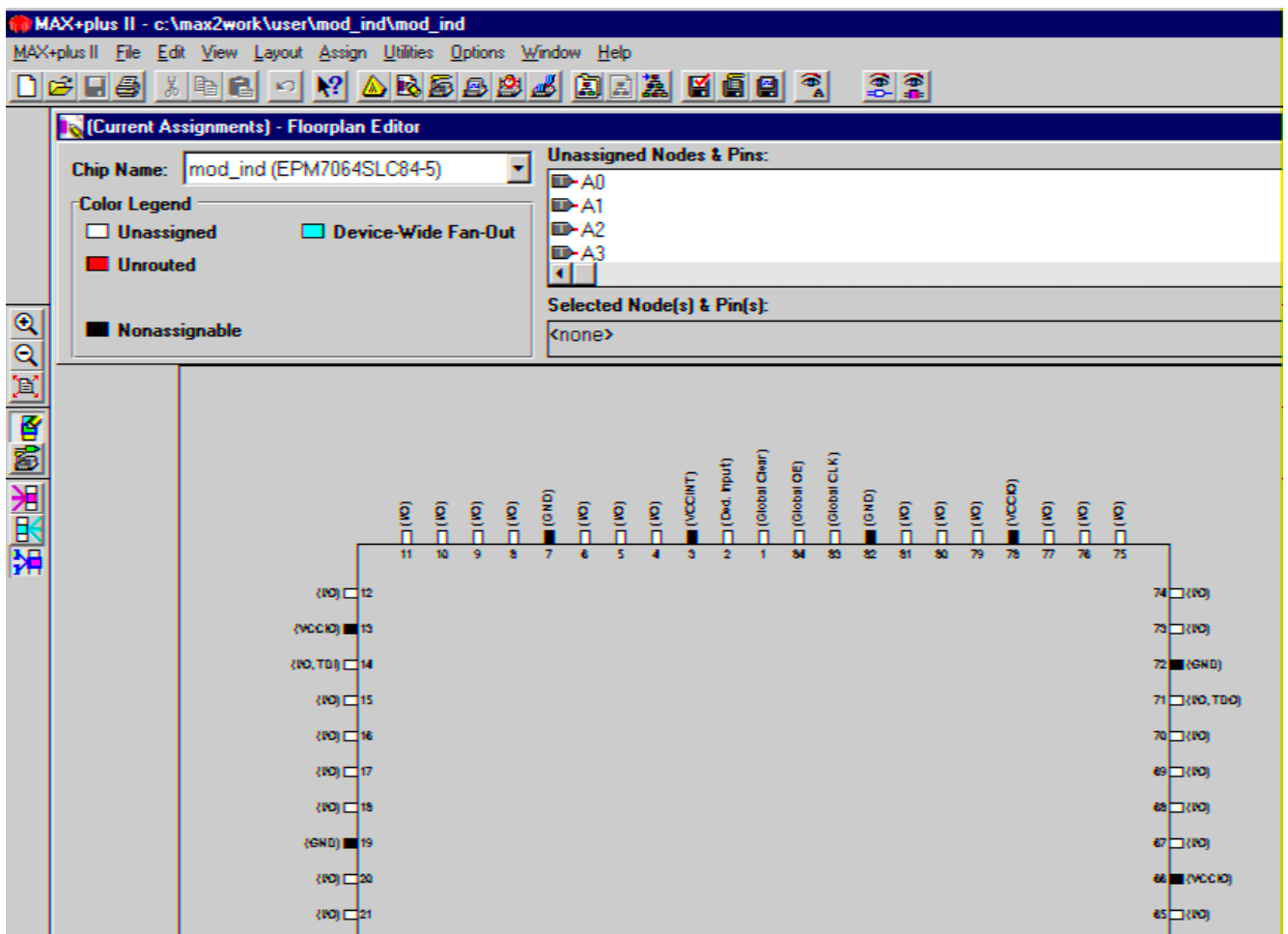


Рис. 9. Окно редактора связей «Floorplan Editor» (при установленном режиме отображения текущих назначений выводов)

С левой стороны окна имеется панель инструментов, среди которых имеется две кнопки: одна из них устанавливает режим отображения результатов компиляции (...last compilation), вторая – режим отображения текущих назначений выводов (...current assignments).

Для назначения или закрепления выводов необходимо мышью перетащить имена выводов, отображаемых в окне «Unassigned Nodes&Pins», на соответствующие номера выводов ПЛИС, а затем перекомпилировать проект. После этого любые изменения, вносимые во внутреннюю структуру проекта, т.е. не связанные с добавлением или удалением выводов, не будут изменять назначение выводов.

## Завершение работы

ПЛИС семейства «MAX 7000S» (а также некоторые другие) имеют встроенную систему программирования ISP, которая использует четыре вывода: #TDI, #TDO, #TMS и #TSK. Эти выходы не должны использоваться для других целей.

Все результаты работы над проектом фиксируются в файле с расширением .RPT, который представляет собой обычный текстовый файл и содержит подробное описание реализованного проекта, в том числе и описание назначения выводов, необходимое для разработки принципиальной схемы целевого устройства. Файл .RPT можно открыть через меню «File» или нажатием соответствующей кнопки на панели инструментов.

Для завершения работы над проектом необходимо провести функциональное моделирование. Для этого необходимо создать исходный файл с расширением .SCF, задать тестовые (эталонные) состояния входов, выбрать проверяемые выходы и запустить «Simulator».

Для создания исходного файла необходимо открыть новый файл, в диалоговом окне «New» отметить «Waveform Editor file», установить в соседнем поле расширение .SCF и нажать «OK», а затем сохранить его через меню «File» основного меню. После этого необходимо связать файл с проектом. Для этого необходимо войти в меню «Node» и выбрать опцию «Enter Nodes from SNF».

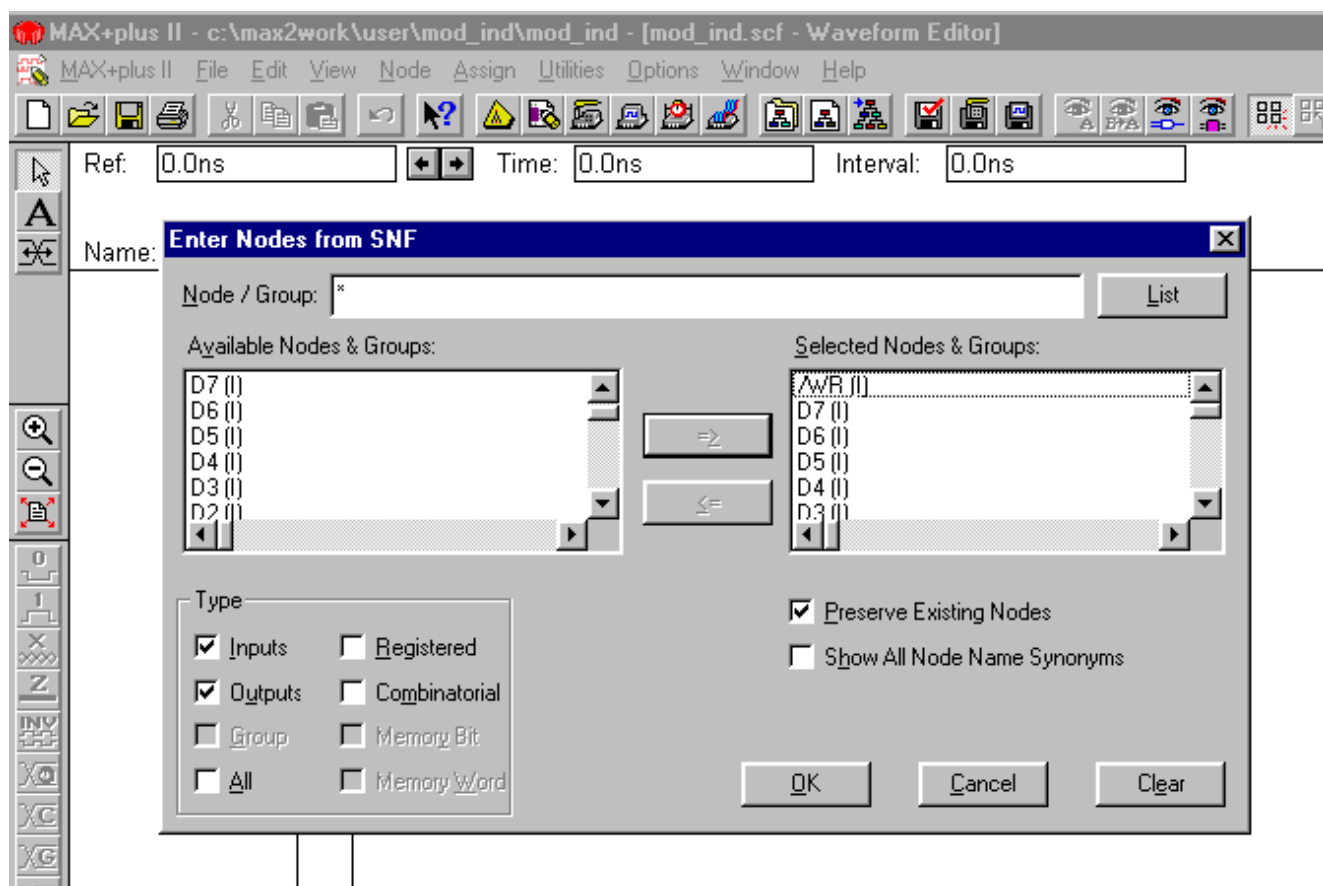
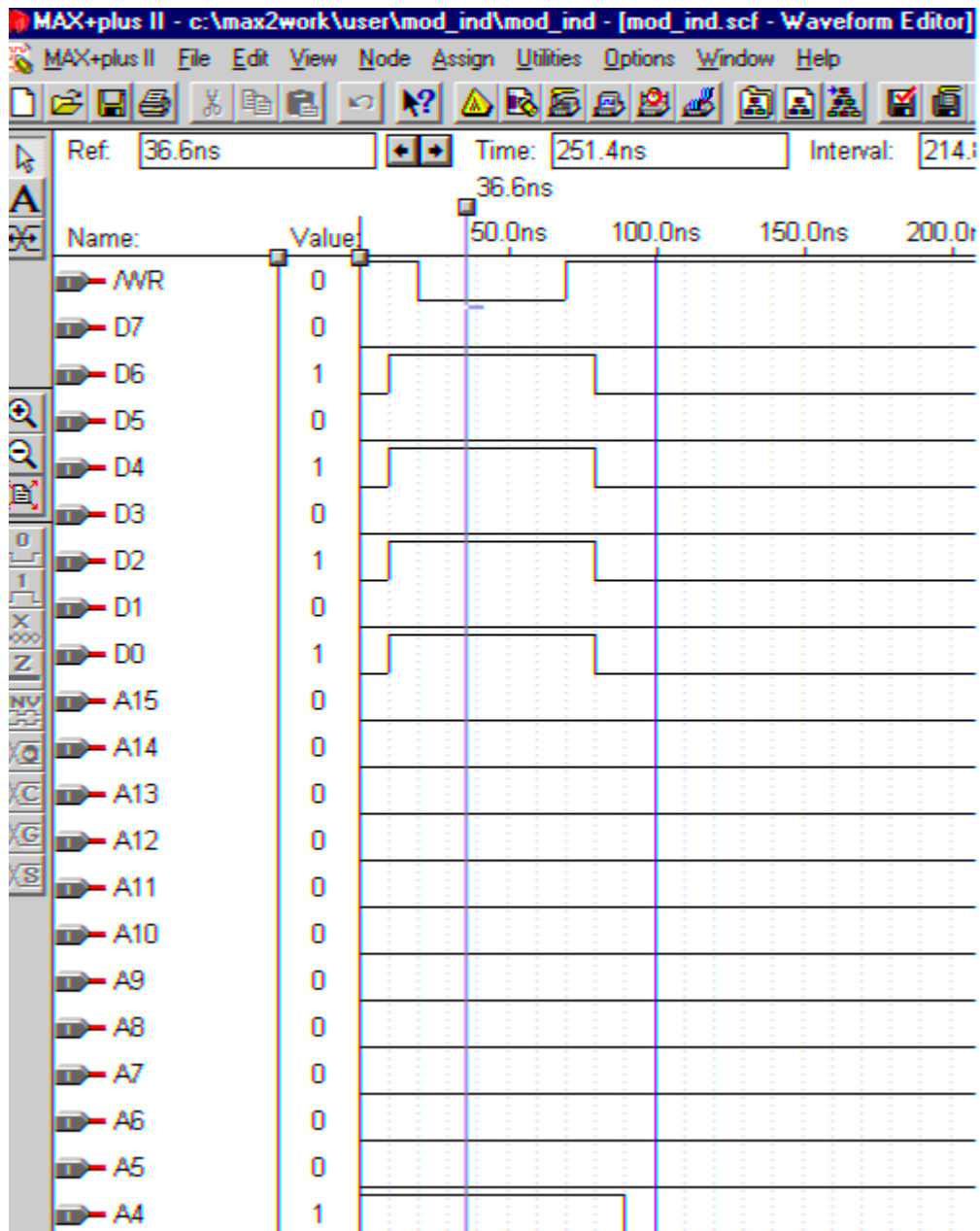


Рис. 10. Диалоговое окно «Enter Nodes from SNF»

После нажатия кнопки «List» в левой панели появляется список доступных узлов («Available Nodes&Groups»), выделенный синим цветом, который необходимо перенести на правую панель («Selected Nodes&Groups»), для чего необходимо нажать кнопку со стрелкой, расположенную между панелями. После нажатия кнопки «OK» в окне редактора появляется готовый шаблон (заготовка) для задания тестовых состояний входов.

Перед началом ввода тестовых состояний необходимо в порядке, описанном выше, задать длительность интервала моделирования и установить шаг сетки с учётом временных параметров реальных сигналов, которые будут подаваться на вход ПЛИС в целевом устройстве.

Сами тестовые состояния входов вводятся так же, как и при создании проекта (см. выше). В данном случае моделируется процесс записи двоичного кода 01010101 по адресу 10h.



**Рис. 11.** Фрагмент исходного файла для моделирования разработанного устройства с заданными состояниями входов

«Simulator» может быть запущен либо через меню «MAX+PLUS II» (в основном меню), либо через панель инструментов, при этом открывается диалоговое окно с кнопками «Start» и «Open SCF». Для начала моделирования необходимо нажать кнопку «Start», а для просмотра результатов – «Open SCF».

Для определения временных соотношений используется приложение «Timing Analyzer», которое запускается так же, как «Simulator». Результаты расчёта временных задержек отображаются в виде таблицы «Delay Matrix»

В заключение ещё раз подчеркнём, что рассмотренный пример является одним из самых простых и иллюстрирует только некоторые приёмы и возможности работы с САПР «MAX+PLUS II». Вместе с тем надеемся, что приведённый пример достаточно хорошо иллюстрирует эффективность системы «MAX+PLUS II» и способен стимулировать стремление заинтересованных специалистов к более глубокому изучению, освоению и использованию этого инструмента в их практической работе.

Желаем творческих успехов!

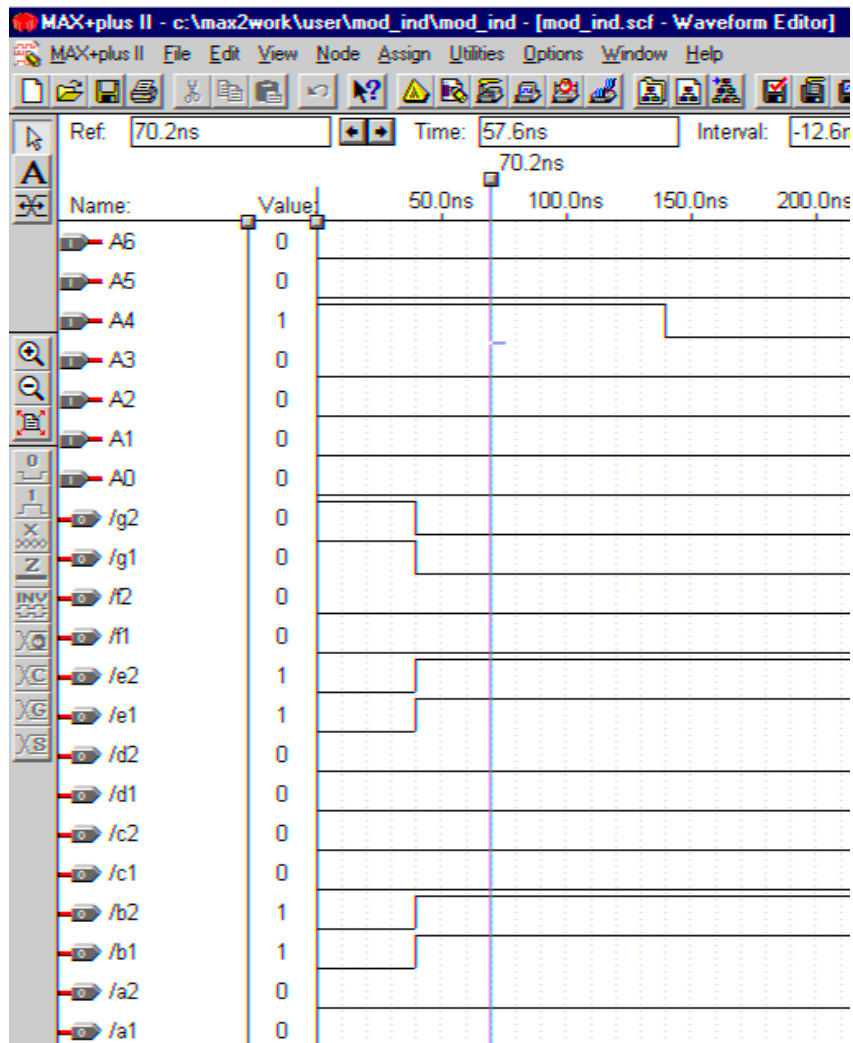


Рис. 12. Фрагмент файла с результатами функционального моделирования

|                            |     | Destination |        |        |        |        |        |
|----------------------------|-----|-------------|--------|--------|--------|--------|--------|
|                            |     | /a1         | /a2    | /b1    | /b2    | /c1    | /c2    |
| S<br>o<br>u<br>r<br>c<br>e | A0  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A1  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A2  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A3  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A4  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A5  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A6  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A7  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A8  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A9  | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A10 | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A11 | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A12 | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A13 | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A14 | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |
|                            | A15 | 10.0ns      | 10.0ns | 10.0ns | 10.0ns | 10.0ns | 10.0ns |

Рис. 13. Фрагмент таблицы «Delay Matrix»