Министерство образования и науки Украины
Севастопольский национальный технический университет

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Лабораторный практикум по дисциплине «Цифровые устройства»

Для студентов специальности 7.090701 дневной и заочной форм обучения

(Часть 1)

Севастополь
2009

УДК 621.396.1

Методические указания «Лабораторный практикум по дисциплине «Цифровые устройства» / Сост. И.Б. Широков, С.Н. Поливкин, М. Синьковский: Ч.1. — Севастополь: Изд-во СевНТУ, 2009. — 50 с.

Целью данной методической разработки является оказание помощи студентам очной и заочной форм обучения в подготовке и проведении лабораторного практикума по дисциплине «Цифровые устройства» а также в получении ими теоретических знаний и практических навыков синтеза логических схем.

Методическое пособие написано на основании программы дисциплины «Цифровые устройства», шифр ИН 004, Киев, 1994 год. Рассмотрено и утверждено на заседании научно-методического семинара кафедры радиотехники (протокол № от июня 2009 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний

Рецензенты: доктор технических наук, профессор Гимпилевич Ю.Б.
кандидат технических наук, доцент Савочкин А.А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Цифровые устройства» разбита на две части и изучается студентами специальности 07090.701 «Радиотехника» дневной и заочной форм обучения в 5-ом и 6-ом семестрах. В данную методическую разработку включены четыре лабораторные работы, выполняемые в первой части дисциплины.

Методические рекомендации по выполнению каждой лабораторной работы содержат следующие разделы: цель работы; теоретическая часть; порядок выполнения работы; содержание отчета; выводы; контрольные вопросы. Для выполнения всех лабораторных работ используется один и тот же макет. Описание лабораторного макета проведено в ознакомительной лабораторной работе № 1.

В основу лабораторного практикума положен фронтальный метод проведения работ, суть которого заключается в том, что группа студентов, разбитая на бригады, выполняет одну и ту же лабораторную работу на различных лабораторных установках. Это позволяет синхронизировать процесс изучения теоретического материала с тематикой лабораторного практикума, а также дает возможность преподавателю по ходу занятия давать пояснения, обсуждать методику эксперимента и результаты со всей группой студентов одновременно. Фронтальный метод позволяет также студентам осуществлять предварительную подготовку к лабораторным занятиям, поскольку их тематика заранее известна.

Бригада состоит, как правило, из 2-3 студентов. В процессе выполнения работы студенты должны вести подробный протокол, в котором излагают весь ход работы, фиксируют результаты экспериментов. После завершения работы преподаватель просматривает полученные данные и подписывает протокол. На основании протокола каждый студент оформляет отчет, который должен иметь титульный лист и содержать следующие разделы: цель работы; синтезируемые схемы и предварительные расчеты; экспериментальные данные (скриншоты синтезированных схем); выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА № 1

ИЗУЧЕНИЕ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ «*MAX+PLUS II*»

1.1. Цель работы

Ознакомление с САПР «*MAX+PLUS II*». Приобретение практических навыков проектирования логических схем, компиляции проектов, программирования ПЛИС и симуляции результатов проектирования.

1.2. Теоретическая часть

1.2.1. Общие положения

Система автоматизированного проектирования «*MAX+PLUS II*» (*Multiple Array Matrix Programmable Logic System* — Пользовательская система программирования логики упорядоченных структур) является полнофункциональной системой проектирования устройств на ПЛИС фирмы «*Altera*». Программное обеспечения этой САПР позволяет осуществить полную разработку проекта, его отладку, а также программирование выбранного типа ПЛИС. Следует отметить, что студенческая версия «*MAX+PLUS II*» (в дальнейшем «система») является бесплатной и свободно распространяется. Основные возможности указанной САПР «*MAX+PLUS II*» приведены в таблице 1.1.

САПР «*MAX+PLUS II*» позволяет использовать разнообразные средства описания проектов с иерархической структурой, осуществлять логический синтез, компиляцию с заданными временными параметрами, выполнять функциональное и временное тестирование (симуляцию проекта).

Создание проекта можно осуществлять с помощью графического редактора «*Graphic Editor*» системы, в виде текстового файла на языке *AHDL* в любом внешнем редакторе или внутреннем «*Text Editor*», а также в виде временной диаграммы, с помощью «*Waveform Editor*». При этом для удобства работы со сложными проектами каждый подпроект может быть представлен в виде символа.

Таблица 1.1 — Основные возможности САПР «*MAX+PLUS II*»

Поддерживаемые устройства	*FLEX* 10K, *FLEX* 8000, *FLEX* 6000, *MAX* 9000, *MAX* 7000, *MAX* 3000, *ACEX* 1K, *EPLDs*,
Средства описания проекта	Схемный ввод, *AHDL*, интерфейс с САПР третьих фирм, встроенный топологический редактор, иерархическая структура проекта, встроенные библиотеки
Средства компиляции проекта	Логический синтез и трассировка, автоматическое обнаружение ошибок
Средства верификации проекта	Временной анализ, функциональное и временное моделирование, анализ сигналов, возможность использования программ моделирования третьих фирм

После установки системы создаются два каталога *maxplus2*, содержащий системное ПО и файлы данных, а также *max2work*, который содержит файлы обучающей программы и примеры. Структуры этих каталогов приведены в таблице 1.2.

Таблица 1.2 — Содержание каталогов САПР

каталог *maxplus*2	Описание
drivers	драйвера устройств ОС
edc	поставляемые фирмой командные файлы
lmf	поставляемые фирмой файлы библиотек, устанавливающие соответствие между логическими функциями пользователя и эквивалентными логическими функциями САПР
max2inc	файлы заголовков с прототипами макрофункций.
max2lib\edif	примитивы и макрофункции, используемые для пользовательских интерфейсов *EDIF*
max2lib\mega_lpm	мегафункции, в том числе библиотека функций параметризированных модулей и файлов заголовков
max2lib\mf	пользовательские макрофункции и устаревшие (логика серии 74XXxxx)
max2lib\prim	стандартные примитивы

Продолжение таблицы 1.2

\ahdl	файлы примеров, иллюстрирующие использование *AHDL*
\chiptrip	все файлы обучающего проекта *chiptrip*, описанного в руководстве САПР
\edif	все файлы примеров, иллюстрирующие особенности *EDIF* в электронном справочнике САПР
\tutorial	информационный файл *read.me* обучающего проекта *chiptrip*
\vhdl	файлы примеров по использованию *VHDL*

1.2.2. Основные модули системы (спадающее меню MAX+PLUS II)

1.2.2. 1. Указатель иерархии (*Hierarchy Display*)

Показывает текущую иерархическую структуру файлов в виде дерева, что может состоять из множества созданных в разных редакторах и свернутых в символы проектов более низких уровней, причем число уровней не ограничивается. При этом можно визуально определить, является ли файл проекта схемным, текстовым или сигнальным. Основной проект (проект наивысшего уровня) должен быть создан в графическом редакторе. Но если проект имеет только один уровень иерархии, то он может быть создан в любом редакторе.

1.2.2.2. Редактор связей (*Floor plan Editor*)

Представляет собой план расположения основных логических элементов. Позволяет вручную распределять выходы ПЛИС (программируемой логической интегрированной схемы), то есть, закреплять ее выходы за конкретными входными и исходными цепями. Также, посредством редактора связей возможно перераспределять некоторые внутренние ресурсы логической схемы.

1.2.2.3. Компилятор (*Compiler*)

Создает файлы, которыми программируются ПЛИС. Обрабатывает логические проекты, разработанные для семейств устройств Altera Classic, MAX3000, MAX5000, MAX7000, MAX9000, FLEX6000, FLEX8000 и FLEX10K. Большинство заданий выполняется автоматически. Однако пользователь может руководить всем процессом компиляции или частично.

1.2.2.4. Симулятор (*Simulator*)

Позволяет проверять правильность логических схем и их внутреннюю синхронизацию. Возможные три режима тестирования: функциональное, временное и тестирование нескольких соединенных между собой устройств. Вместе с редактором временных диаграмм используется для функционального моделирования проекта с целью проверки правильности работы логики.

1.2.2.5. Анализатор временных параметров (*Timing Analyzer*)

Проверяет работу проектируемой логической цепи после того, как он был синтезирован и оптимизирован компилятором. Позволяет оценить задержки распространения сигналов, что возникают в схеме.

1.2.2.6. Программатор (*Programmer*)

Позволяет программировать, конфигурировать, проводить верификацию и опробовать ПЛИС фирмы Altera.

1.2.3. Редакторы системы MAX+PLUS II

1.2.3.1. Графический редактор(*Grafic Editor*)

Предназначен для создания проекта в виде схемы соединений символов логических элементов, что берутся из библиотек пакета.

1.2.3.2. Символьный редактор(*Symbol Editor*)

Позволяет редактировать существующие и создавать новые символы. Любой проект после компиляции может быть свернут в символ, помещенный в библиотеке), но использован, как элемент в другом проекте.

1.2.3.3. Текстовый редактор(*Text Editor*)

Позволяет создавать и редактировать текстовые файлы проекта, которые содержат описания логики проекта языком *AHDL* (*Altera Hardware Description Language*), близких к нему языках или типу *VHDL* или *Verilog*.

1.2.3.4. Редактор временных диаграмм(*Waveform Editor*)

Некоторые авторы называют этот редактор сигнальным. Он выполняет двойную функцию: на этапе синтеза проекта обеспечивает создание логики в виде диаграмм (эпюр) состояний входов и выходов, а на этапе моделирования создания диаграмм тестовых (эталонных) входных состояний устройства, которое моделируется.

1.2.4. Проектирование логических схем

Рассмотрим процесс работы с САПР «*MAX+PLUS II*». Внешний главного меню и панели инструментов проекта показан на рис. 1.1. Описание панели инструментов приведено в таблице 1.3.

Рис. 1.1 — Функциональный состав панелей системы

Таблица 1.3 — Описание панели инструментов «*MAX+PLUS II*»

№	Описание	№	Описание
1	Создать новый файл	12	Компилятор
2	Открыть существующий файл	13	Симулятор
3	Сохранить изменения в текущий файл	14	Анализатор временных диаграмм
4	Распечатать текущий файл или содержимое окна	15	Программатор
5	Вырезать фрагмент	16	Открыть проект
6	Скопировать фрагмент	17	Изменить название проекта на имя текущего файла
7	Вставить фрагмент	18	Показать главный файл проекта
8	Отмена предыдущего действия	19	Сохранить все открытые компилятором файлы и проверить текущий проект на синтаксические ошибки
9	Контекстно-зависимая помощь	20	Сохранить все открытые файлы проекта и запустить компилятор
10	Менеджер проекта	21	Сохранить все открытые входные файлы симулятора и запустить симулятор
11	Редактор графики и символов		

Рассмотрим основные принципы работы с этой системой. Для создания нового проекта создайте новый файл проекта (*File>New* или пиктограмма 1 на рис. 1.1). При этом система спросит, какой тип файла создать (см. рис. 1.2).

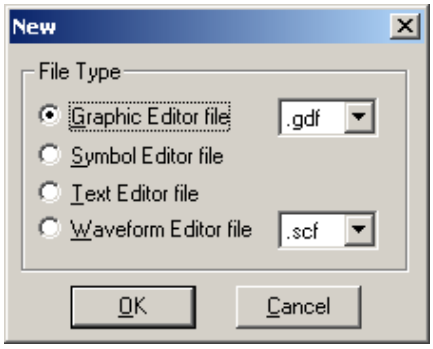


Рис. 1.2 — Выбор типа файла

Для работы с проектом с помощью графического редактора «*Graphic Editor*» выбираем тип файла *.gdf*. После нажатия кнопки ОК система создает пустое окно с рабочим файлом графического редактора. Ввод символа осуществляется путем двойного щелчка мыши по пустому полю графического проекта. При этом появляется окно ввода символа (см. рис. 1.3).

Поскольку принятые в системе графические обозначения логических элементов отличаются от стандартных отечественных, в Приложении приведена таблица соответствия условных графических обозначений.

Выбор символа осуществляется путем указания его сокращенного названия в пункте «*Symbol name*». Перечень необходимых для выполнения работы стандартных символов приведен в Приложении. После ввода схемы необходимо указать ее входы и выходы. В графическом редакторе по умолчанию они имеют названия «*input* » и «*output*» соответственно (см. рис. 1.4).

Для улучшения восприятия схемы следует назначать всем её входам и выходам осмысленные имена. Для изменения названия следует дважды щелкнуть левой кнопкой мыши по текущему названию порта (например *PIN_NAME*) и изменить название (см. рис.1.5). При этом следует использовать только латинские буквы, цифры и знак нижнего подчеркивания.

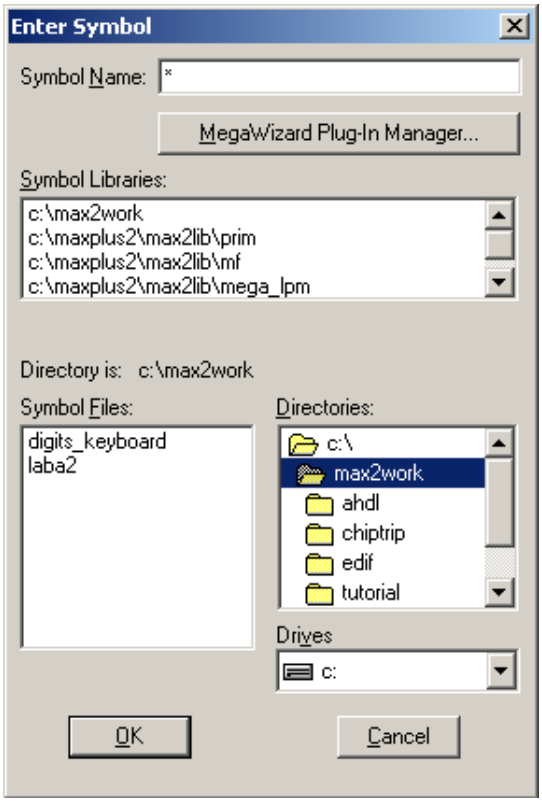


Рис. 1.3 — Окно выбора символов

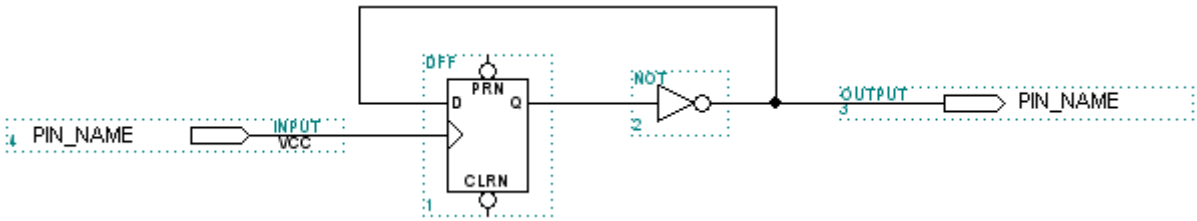


Рис. 1.4 — Ввод принципиальной схемы

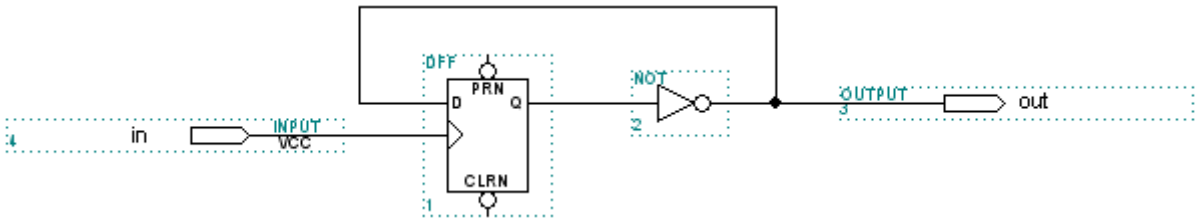


Рис. 1.5 — Изменение имен входных и выходных портов

При необходимости любой элемент схемы может быть использован как отдельный элемент. Для выполнения этой операции необходимо нарисовать желаемую схему, дать названия входным и выходным портам и сохранить ее с названием, соответствующим названию нового элемента. После этого с помощью команды (*File>Create Default Symbol*) сохраняем схему в виде элемента. При этом в качестве название нового элемента будет соответствовать названию файла схемы, из которой

он был создан. Входы нового элемента будут расположены слева, выходы — справа. Для использования нового элемента достаточно набрать его имя, аналогично стандартным элементам *MAX+PLUS II*. Двойной щелчок левой кнопкой мыши на таком элементе откроет его внутреннюю схему. Для внесения изменения в схему элемента следует открыть его, внести необходимые изменения, после чего сохранить их (*File>Save* или пиктограмма 3 на рис. 1.1), после чего повторно создать символ (*File>Create Default Symbol*). После этого можно закрывать внутреннюю схему символа. При создании нового элемента можно использовать другие элементы. При изменении элемента, для обновления проекта следует использовать команду **(***Symbol>Update Symbol***)**.

1.2.5. Компиляция проекта

Перед компиляцией следует сохранить файл проекта (File>Save или пиктограмма 3 на рис. 1.1). Кроме того, нужно указать, с каким файлом будет работать компилятор. При компиляции текущего файла следует выполнить следующую последовательность действий: File>Project>Set Project to Current File. В противном случае будет произведена компиляция проекта, который был текущим в прошлом сеансе работы с системой.

Для компиляции проекта следует вызвать окно компилятора из меню (*MAX+PLUS II > Compiler* или пиктограмма 12 на рис. 1.1) (рис. 1.6).

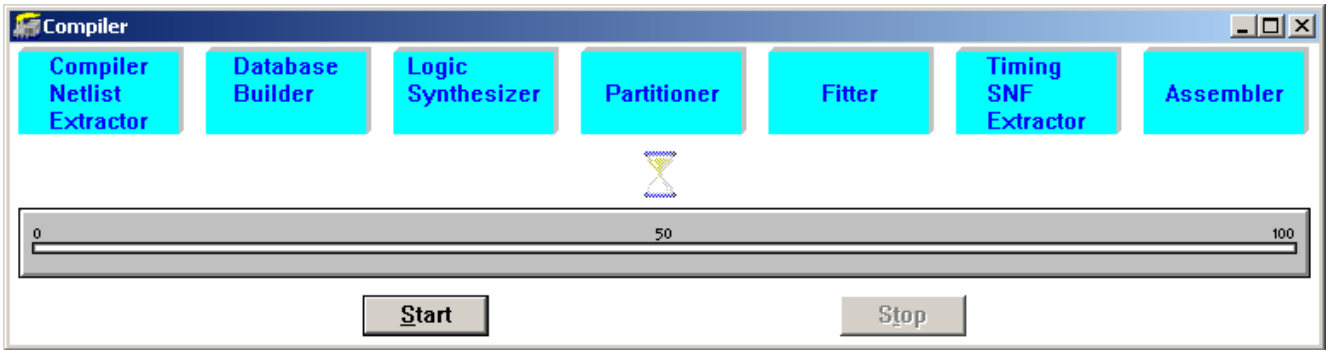


Рис. 1.6 — Окно компилятора проекта

Для начала компиляции проекта необходимо нажать кнопку *Start* в окне компилятора. В процессе компиляции индикатор будет последовательно показывать динамику процесса, что может помочь при возникновении ошибок и выдавать сервис-

ные сообщения. При успешном завершении компиляции будет выдано соответствующее сообщение (см. рис.1.7).

С помощью окна *Messages*, в котором при компиляции отображаются сообщения системы, предупреждения и ошибки, можно вызвать встроенную справочную систему. Кроме того, это окно позволяет перейти на фрагмент схемы, вызвавший появление соответствующего сообщения или ошибки. Для этого достаточно выполнить двойной щелчок мыши по соответствующему сообщению.

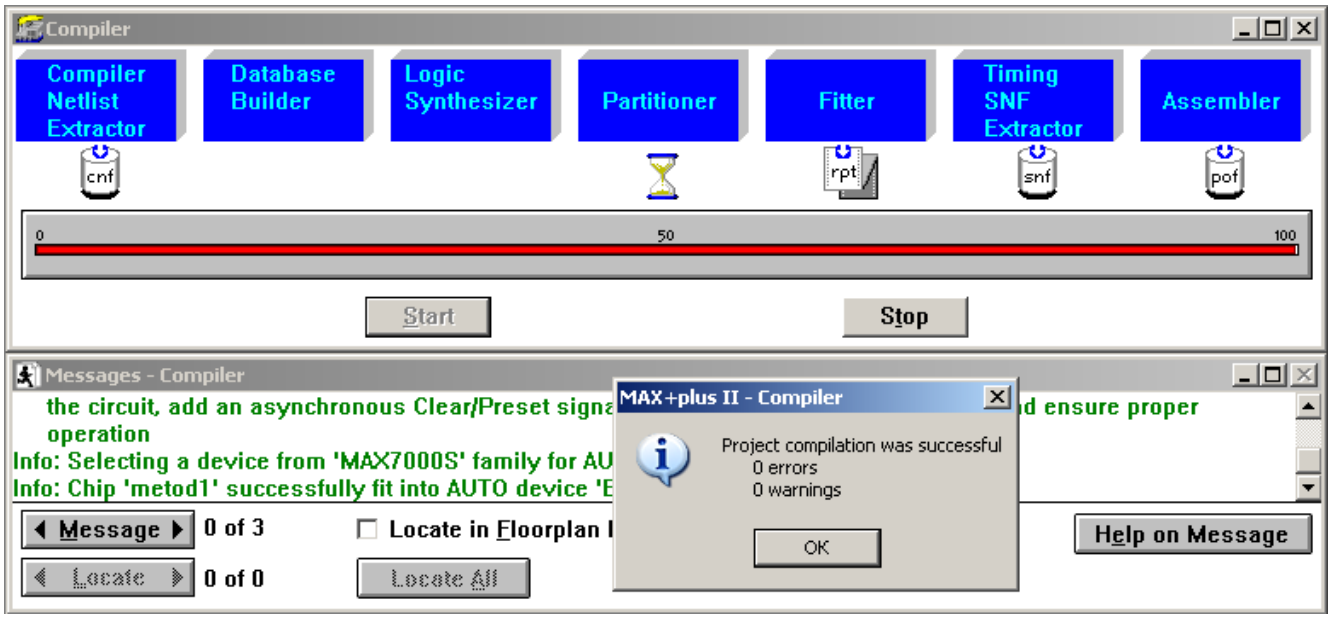


Рис. 1.7 — Сообщения системы в случае успешной компиляции

1.2.6. Проверка работоспособности проекта

После завершения компиляции необходимо проверить работоспособность введенной схемы. Один из наиболее наглядных способов заключается в анализе временных диаграмм входных и выходных сигналов схемы. Для получения временных диаграмм текущей схемы следует вызвать окно редактора графиков (*MAX+PLUS II* > *Waveform Editor*) (см. рис.1.8). Для наблюдения за интересующими нас входами и выводами схемы следует с помощью правой кнопкой мыши в графах *Name* и *Value* вызвать контекстное меню, в котором необходимо выбрать пункт «*Insert Node*» для вставки одного вывода, после чего ввести его имя либо выбрать из списка в раскрывшемся окне выбора (рис. 1.9).

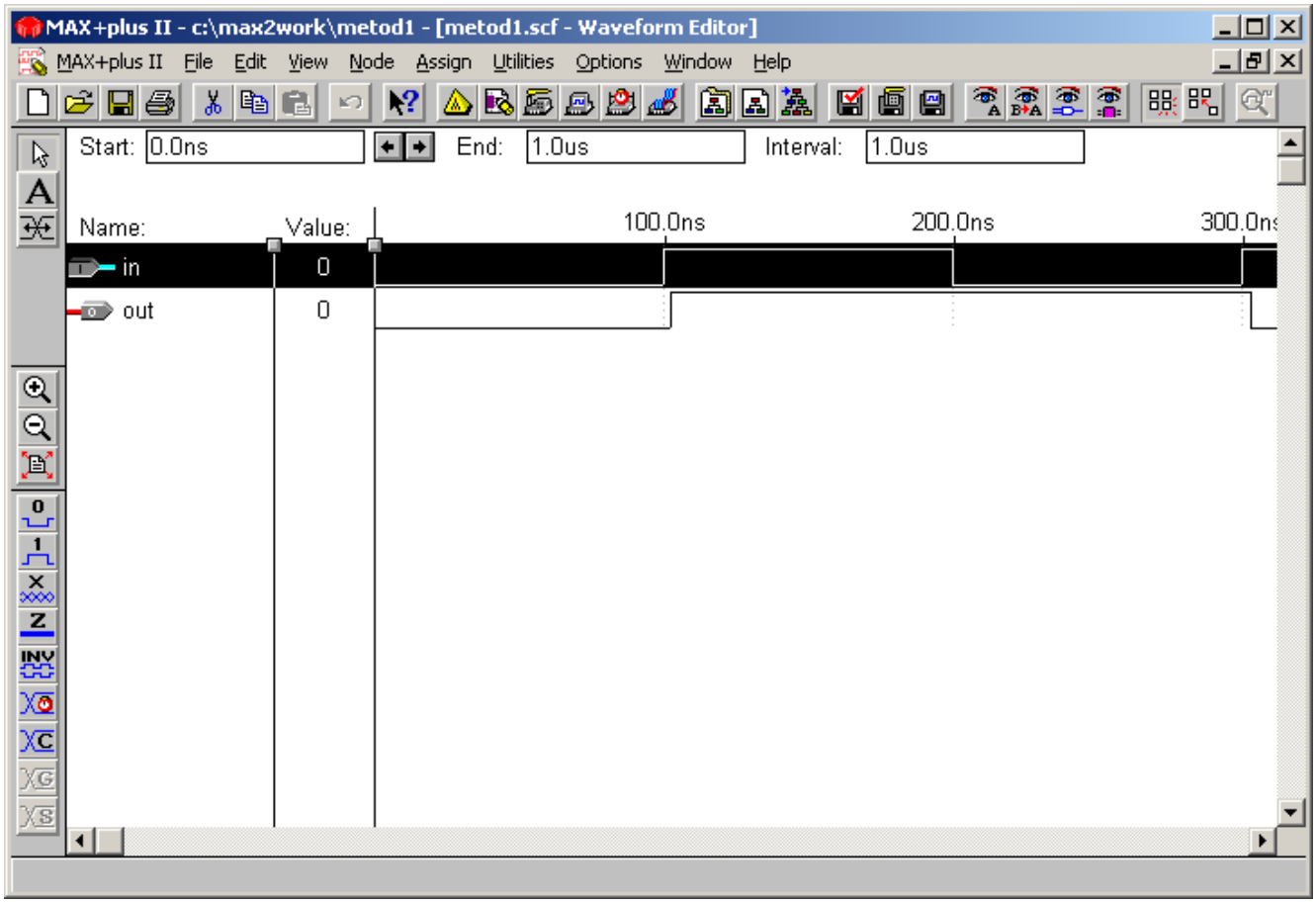


Рис. 1.8 — Внешний вид окна редактора графиков

При выборе пункта контекстного меню «*Enter Nodes from SNF*» можно добавить сразу несколько выводов в таблицу (рис. 1.10). Для этого необходимо с помощью кнопки «*List*» вывести список выводов, соответствующих выбранному типу, отметить нужные, переместить в окно выбранных элементов и подтвердить свой выбор путем нажатия кнопки «ОК».

В результате, в окне редактора графиков появятся выбранные входы и выходы схемы. На входы схемы нужно подать тестовые сигналы. Вкладка с различными типами тестовых сигналов показана в левой части окна редактора графиков (рис. 1.8).

После подачи на входы схемы необходимых тестовых сигналов следует сохранить файл редактора графиков, после чего вызвать окно настроек симулятора, показанное на рис. 1.11 (*MAX+PLUS II* > *Simulator* пиктограмма 13 на рис. 1.1).

С помощью окна настроек симулятора можно задать временной интервал (*Start Time – End Time*), в котором будут определяться состояния выходных сигналов, а также определить дополнительные настройки.

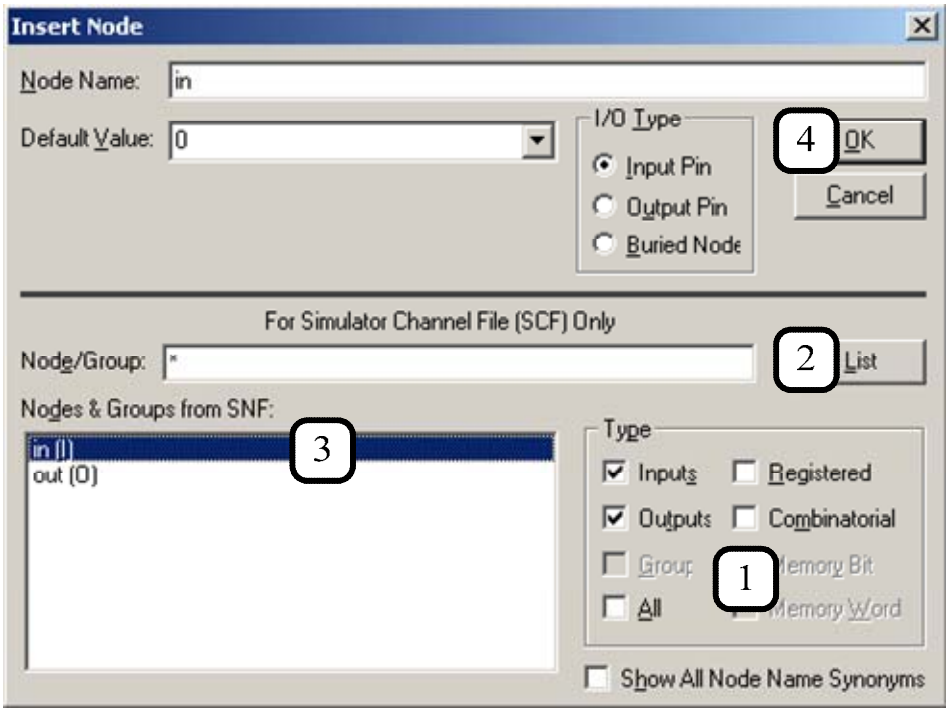


Рис.1. 9 — Последовательность действий при осуществлении добавки одиночного вывода в список

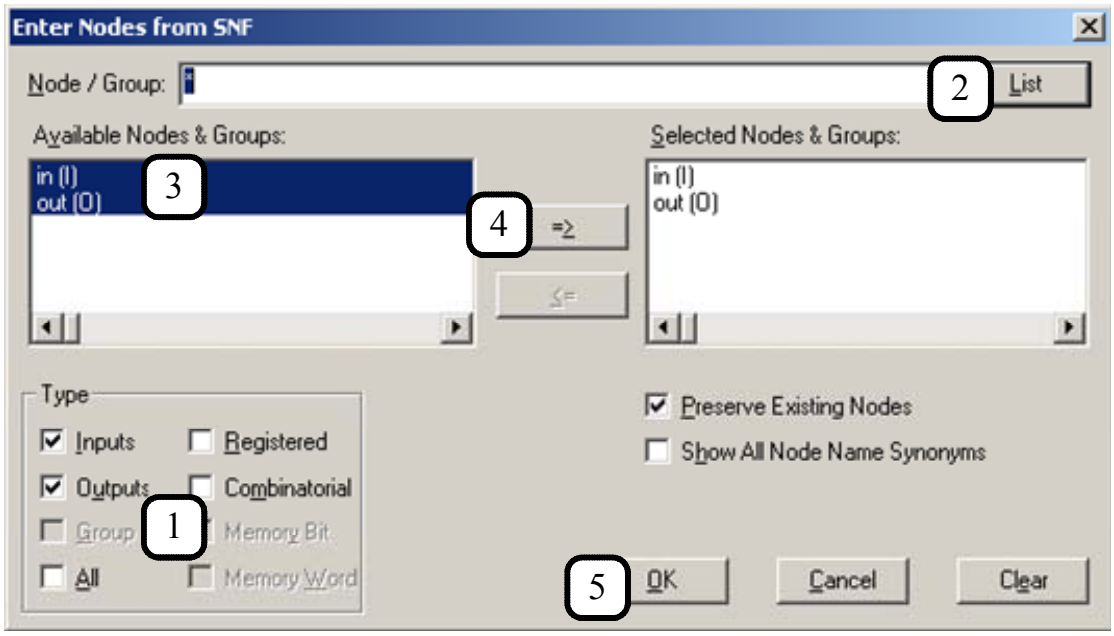


Рис.1.10 — Последовательность действий при осуществлении добавки группы выводов в список

При выборе временного интервала симуляции следует учитывать, что величина «*End Time*» не должна быть больше максимального значения, установленного в редакторе графиков (по умолчанию 1 мкс), в противном случает система выдаст сообщение об ошибке. Для анализа графиков на временном интервале, превышающем 1 мкс следует изменить значение «*End Time*» (*File* > *End Time*).

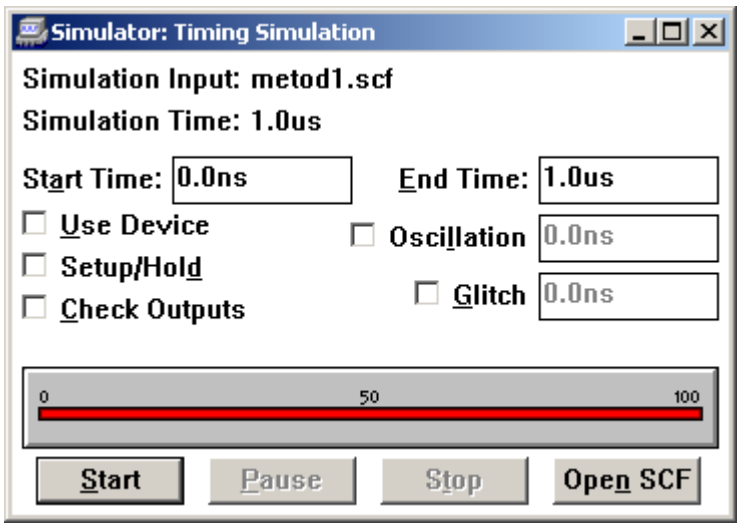


Рис. 1.11 — Окно симулятора

При отсутствии ошибок и корректном задании временного интервала симулятор определит состояния выходов схемы с учетом задержек сигналов в микросхемах и построит соответствующие графики в окне редактора графиков (см. рис. 1.8).

1.2.7. Назначение выводов ПЛИС

Для дальнейшей работы с проектом необходимо определить тип используемой ПЛИС и указать, какие ее выводы будут использоваться в проекте. *MAX+PLUS II* позволяет выполнить эти операции на любом этапе создания проекта, однако следует помнить, что после изменения типа используемого кристалла, а также переназначении выводов, следует заново скомпилировать проект. По умолчанию система самостоятельно выбирает тип ПЛИС, ресурсов которого достаточно для проекта. В стенде «РТ-2007» используется ПЛИС семейства *MAX*7000S типа *EPM*7064*SLC*44. Выбор используемой ПЛИС осуществляется в меню *Assign* (*Assign* > *Device*). При выборе кристалла сначала нужно указать его семейство, после чего выбрать тип.

Привязку выводов, используемых в проекте к физическим выводам ПЛИС можно выполнить с помощью команды (*Assign* > *Pin*/*Location*/*Chip*) Внешний вид окна привязки показан на рис. 1.12. Назначение выводов ПЛИС стенда приведено в приложении.

Для осуществления привязки вывода ПЛИС к порту схемы следует указать название порта в разделе «*Node Name*», после чего установить номер вывода ПЛИС, которому должен соответствовать этот порт (раздел «*Chip Resource*»). После выполнения описанных выше операций и подтверждения текущей операции привязки

(кнопка «*Add*») в окне привязки появится информация о имени порта в проекте, текущем кристалле, типе вывода и номере вывода ПЛИС. Аналогичным образом следует привязать все входные и выходные порты, используемые в проекте к выводам ПЛИС. После компиляции просмотреть и, в случае необходимости, скорректировать результаты привязки можно в окне редактора компоновок (*MAX+PLUS II > Floorplan editor*).

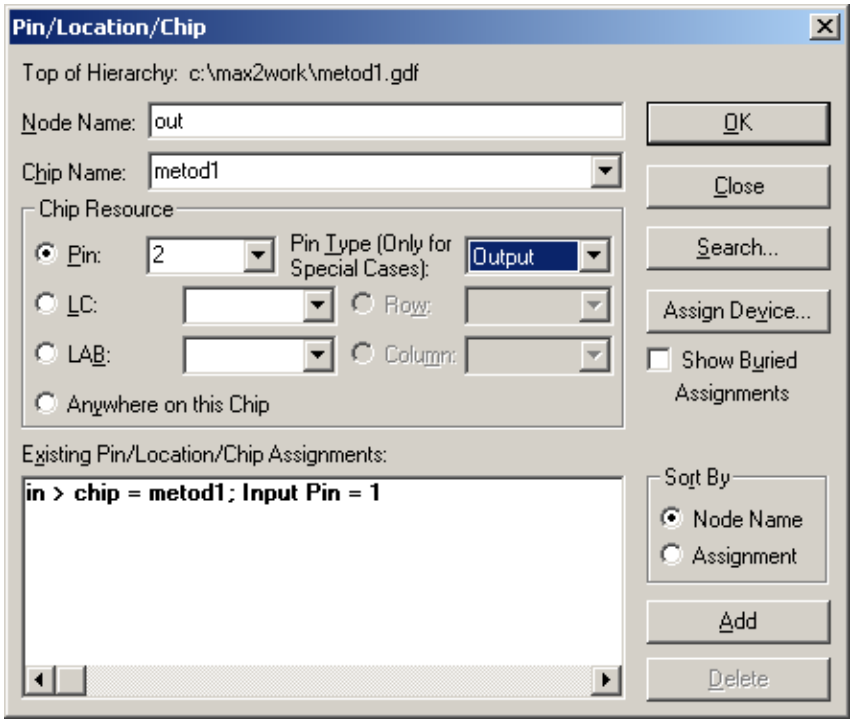


Рис. 1.12 — Окно привязки выводов

1.2.8. Программирование ПЛИС

После успешной компиляции проекта можно осуществить программирование ПЛИС, после которого она будет вести себя, подобно схеме этого проекта. Для выполнения программирования следует вызвать окно программатора (*MAX+PLUS II > Programmer*). Внешний вид окна программатора показан на рис. 1.13.

Для осуществления программирования при подключенном стенде «РТ-2007» достаточно нажать кнопку «*Program*», после чего ПЛИС будет сконфигурирована в соответствии с текущим проектом.

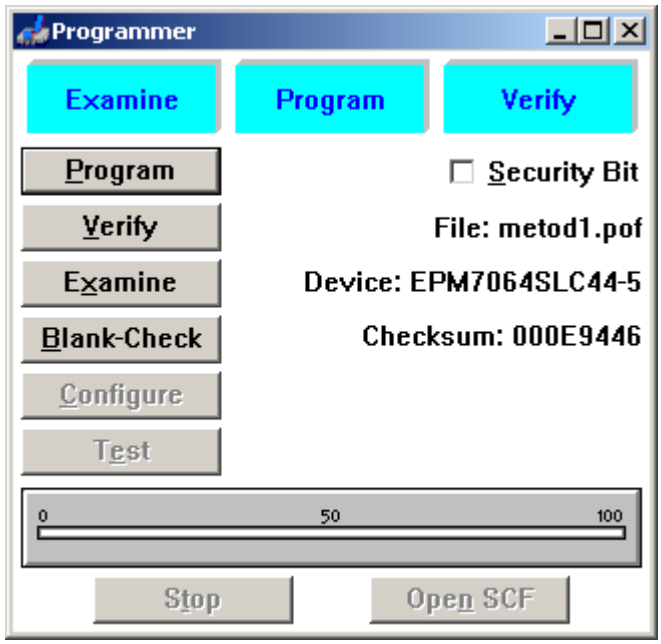


Рис. 1.13 — Окно программатора

1.3. Описание лабораторной установки

1.3.1. Расположение и назначение элементов стенда «РТ-2007»

Лабораторный стенд представляет собой функционально законченное устройство, содержащее в своем составе собственно программируемую логическую интегральную микросхему семейства *MAX*7000S типа *EPM*7064*SLC*44, программатор, периферийные устройства и блок питания. Внешний вид стенда и расположение его составных частей показан на рис. 1.14.

Назначение выводов ПЛИС в стенде «РТ-2007» приведены в таблице 1.4.

Световые индикаторы *HL*1 и *HL*2 представляют собой семисегментные (*A, B, C, D, E, F, G*) светодиодные индикаторы. Для зажигания соответствующего сегмента на назначенном выводе ПЛИС необходимо сформировать лог. 1 .

Для зажигания линейки отдельных светодиодов *LED*1… *LED*8, на назначенном выводе ПЛИС необходимо сформировать лог. 1 .

Кнопки *SA*1 и *SA*2 позволяют при ее нажатии формировать на соответствующем входе ПЛИС лог. 0 (стационарное состояние — лог. 1).

Переключатели *SW*1… *SW*4 позволяют устанавливать на соответствующих входах ПЛИС долговременно лог. 0 или лог. 1.

Тактовый генератор формирует на входе *CLOCK* ПЛИС последовательность импульсов с частотой следования около 1 кГц.

Программирование ПЛИС осуществляется посредством программатора *Byte-Blaster*, который подключается к *LPT*-разъему компьютера.

Таблица 1.4 —Назначение выводов ПЛИС в стенде «РТ-2007»

Назначение	Номер вывода	Направление	Примечание	Назначение	Номер вывода	Направление	Примечание
*A*1	4	выход	индик.1	*LED*1	18	выход	светодиод
*B*1	5	выход	индик.1	*LED*2	19	выход	светодиод
*C*1	31	выход	индик.1	*LED*3	20	выход	светодиод
*D*1	29	выход	индик.1	*LED*4	21	выход	светодиод
*E*1	27	выход	индик.1	*LED*5	24	выход	светодиод
*F*1	12	выход	индик.1	*LED*6	25	выход	светодиод
*G*1	6	выход	индик.1	*LED*7	26	выход	светодиод
*A*2	41	выход	индик.2	*LED*8	28	выход	светодиод
*B*2	39	выход	индик.2	*SA*1	8	вход	кнопка
*C*2	37	выход	индик.2	*SA*2	9	вход	кнопка
*D*2	33	выход	индик.2	*SW*1	11	вход	переключат.
*E*2	34	выход	индик.2	*SW*2	14	вход	переключат.
*F*2	40	выход	индик.2	*SW*3	17	вход	переключат.
*G*2	36	выход	индик.2	*SW*4	16	вход	переключат.
				CLOCK	43	вход	тактовый генератор

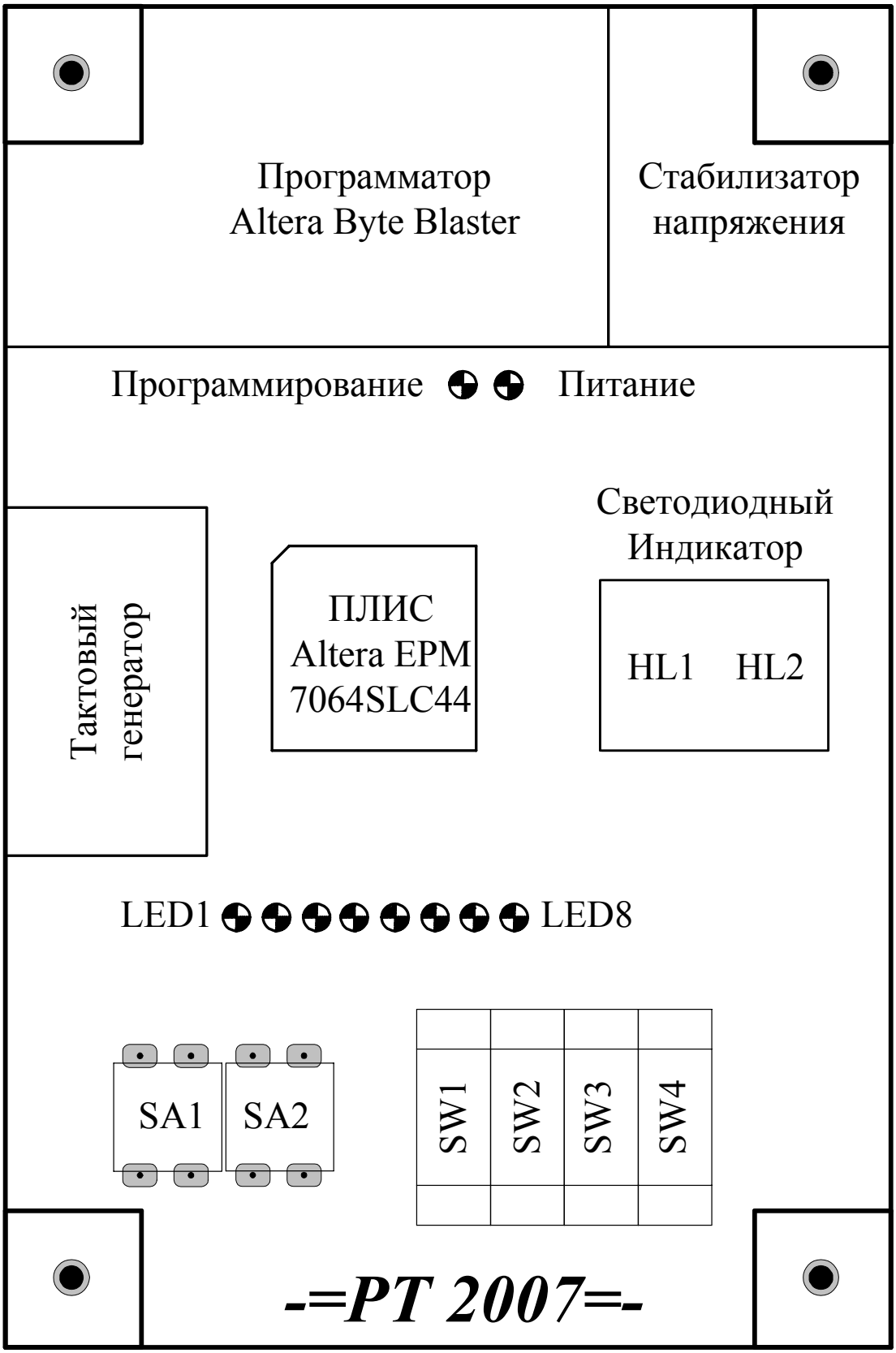
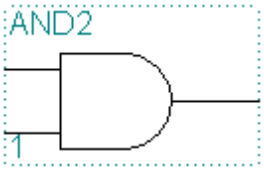
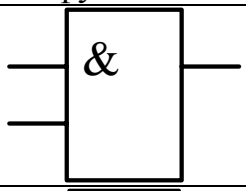
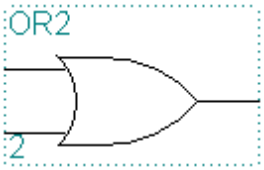
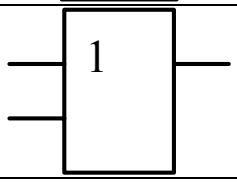
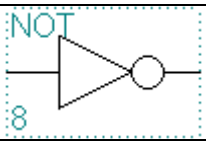
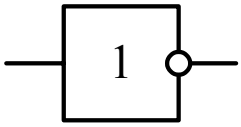
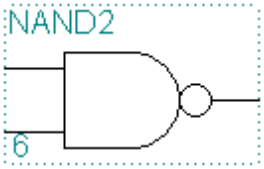
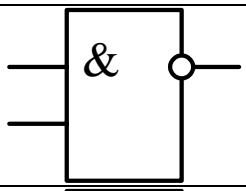
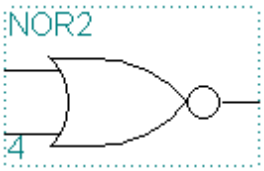
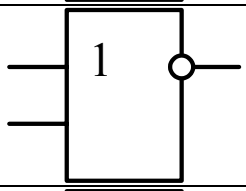
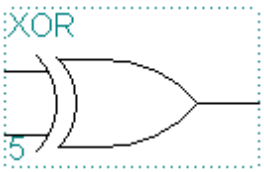
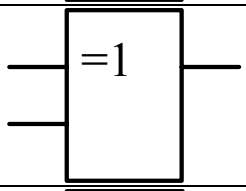
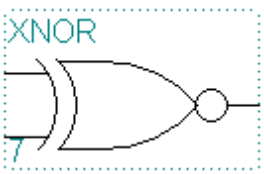
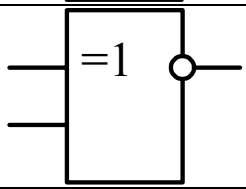


Рис. 1.14 — Внешний вид стенда

1.3.2. Обозначения основных логических элементов в САПР

Соответствие обозначений основных логических элементов, используемых в САПР *MAX+PLUS II* и принятым в ДСТУ приведено в таблице 1.5.

Таблица 1.5 — Соответствие обозначений основных логических элементов

Название элемента	Графическое обозначение функции в *MAX+PLUS II*	Стандартное графическое обозначение функции	Логическая функция
2И (Конъюнктор)			$y = X1 \cdot X2 = = \overline{\overline{X1} + \overline{X2}}$
2ИЛИ (Дизъюнктор)			$Y = X1 + X2 = = \overline{\overline{X1} \cdot \overline{X2}}$
НЕ (Инвертор)			$Y = \overline{X}$
2И-НЕ			$Y = \overline{X1 \cdot X2} = = \overline{X1} + \overline{X2}$
2ИЛИ-НЕ			$Y = \overline{X1 + X2} = = \overline{X1} \cdot \overline{X2}$
Исключающее ИЛИ			$Y = X1 \cdot \overline{X2} + + \overline{X1} \cdot X2$
Исключающее ИЛИ-НЕ			$Y = X1 \cdot X2 + + \overline{X1} \cdot \overline{X2}$

1.4. Порядок выполнения работы

1.4.1. Создайте в своей рабочей папке директорию *sample*, и перепишите к ней все файлы с *C:\max2work*\PT2007__core, что имеют имя *dec_7seg* (декодер 7-сегментного дисплея), *debourtce* (антидребезг) и *clk_div* (входной делитель частоты).

1.4.2. Откройте ПП *MAX+PLUS II*. Откройте окно *New* в меню *File*. Выберите пункт *Graphic Editor file* и нажмите *OK*. Откроется пустое окно *Untitled*-I графического редактора.

1.4.3. Из контекстного меню (вызывается правой клавишей мыши в окне редактора) откройте окно *Enter Symbol*. В графе *Symbol Name* наберите ***INPUT*** (без кавычек) и нажмите *OK*. На экране появится одноименный элемент. Если положение элемента на рабочем поле не специфицировано нажатием левой кнопки мыши, элемент помещается в центр рабочего поля. Перемещать элемент можно традиционным для *Windows* методом.

1.4.4. Так же введите элемент ***OR2*** (элемент 2ИЛИ). Обратите внимание, что в верхнем левом угле элемента указано его имя. Расположите элемент ***OR2*** напротив предыдущего элемента, перетаскивая его курсором мыши. Наведите курсор мыши на правый конец элемента ***INPUT***, его изображение изменится на плюс. Нажав и удерживая клавишу мыши проведите черту от выхода ***INPUT*** к входу элемента ***OR2***. Линию связи можно сгибать, но только один раз.

1.4.5. Элемент ***INPUT*** имеет служебное имя, что указано сбоку от элемента. Замените его на ***IN*** командой *Edit Pin Name* из контекстного меню элемента.

1.4.6. Пользуясь полученными навыками, дорисуйте схему изображенную на рис. 1.15.

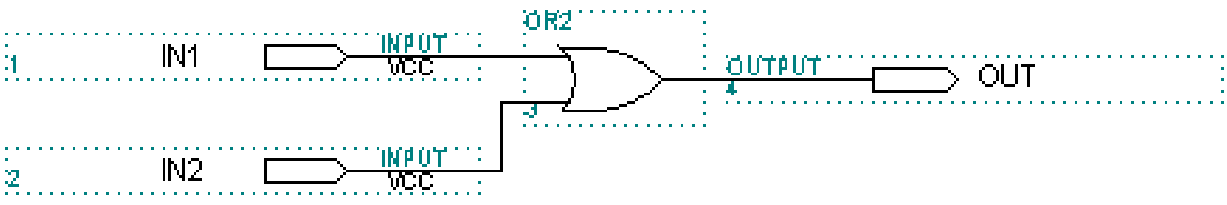


Рис. 1.15 — Элемент 2 ИЛИ

1.4.7. Командой *Save As* (меню *File*) запомните созданный файл с именем *sample*_1.

1.4.8. Командой *Set Project to Current File* (пункт *File* меню *Project*) назначьте текущий файл главным файлом проекта.

1.4.9. Откройте компилятор (пункт *Compiler* меню *MAX+PLUS* II) и откомпилируйте проект нажав на *Start*. Если компиляция завершилась с ошибками, исправьте их, сверяясь с схемой и перекомпилируйте проект.

1.4.10. Откройте редактор графических форм сигналов (пункт *Waveform Editor* меню *MAX+PLUS* II). Откройте окно *Enter Nodes from SNF* (меню *Node*). Нажмите на *List*. В окне *Avaible Nodes & Groups*: появится перечень входных и выходных цепей, доступных для анализа. Перенесите все цепи в окно *Selected Nodes & Groups*:, нажав "=>". Нажмите *OK*.

1.4.11. Наведите курсор на участок диаграммы цепи ***IN2***, который находится между отметками 200.0ns и 300.0ns. Выделите этот участок, нажав левую клавишу мыши и переместив ее вдоль диаграммы. На левой инструментальной панели нажмите на кнопку с изображением единицы. Вы увидите, что на выделенном промежутке времени на входную цепь будет подан сигнал "логическая единица". Так же создайте временные диаграммы для всех входных сигналов согласно рис. 4.

1.4.12. Посредством команды *Save As* (меню *File*) запомните созданный файл с именем по умолчанию.

1.4.13. Откройте окно *Simulator* (меню *MAX+PLUS* II) и проверьте правильность функционирования проекта, нажав на *Start*. Если симуляция завершена без ошибок, закройте симулятор и перейдите к окну сигнального редактора, в котором будут изображены временные диаграммы входных и выходного сигналов после симуляции.

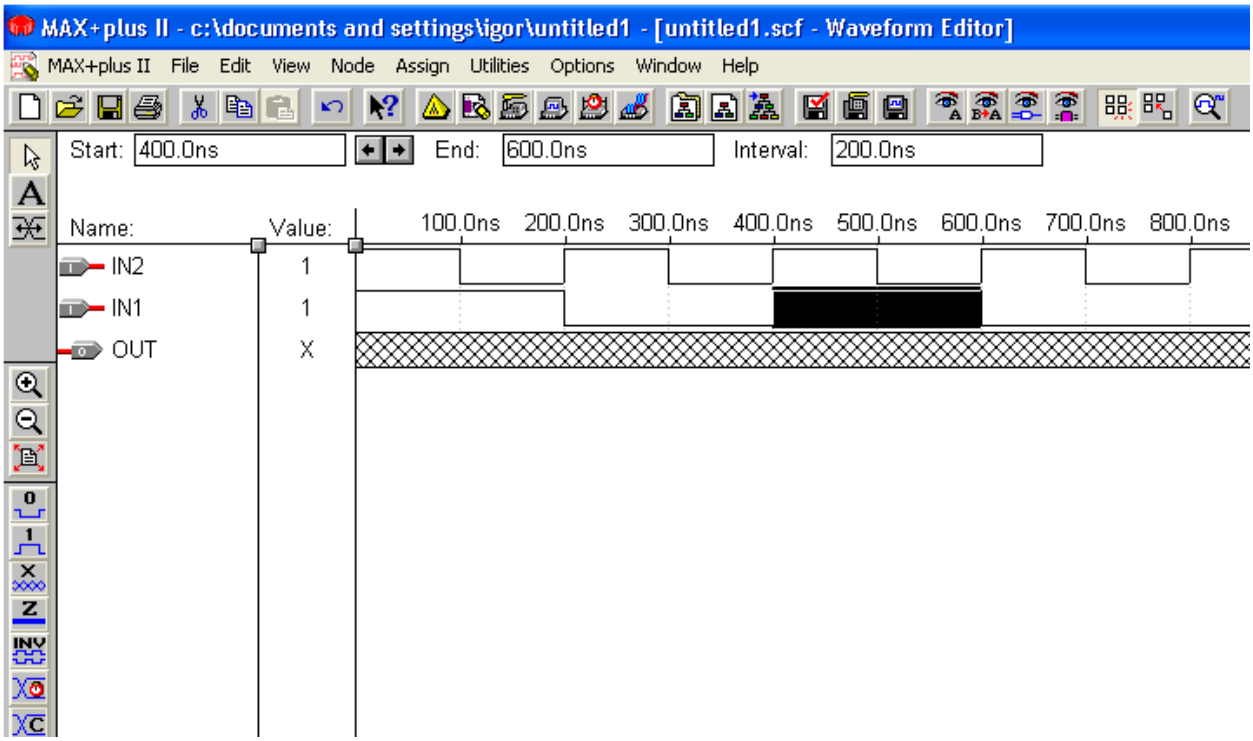


Рис. 1.16 — Временные диаграммы входных сигналов

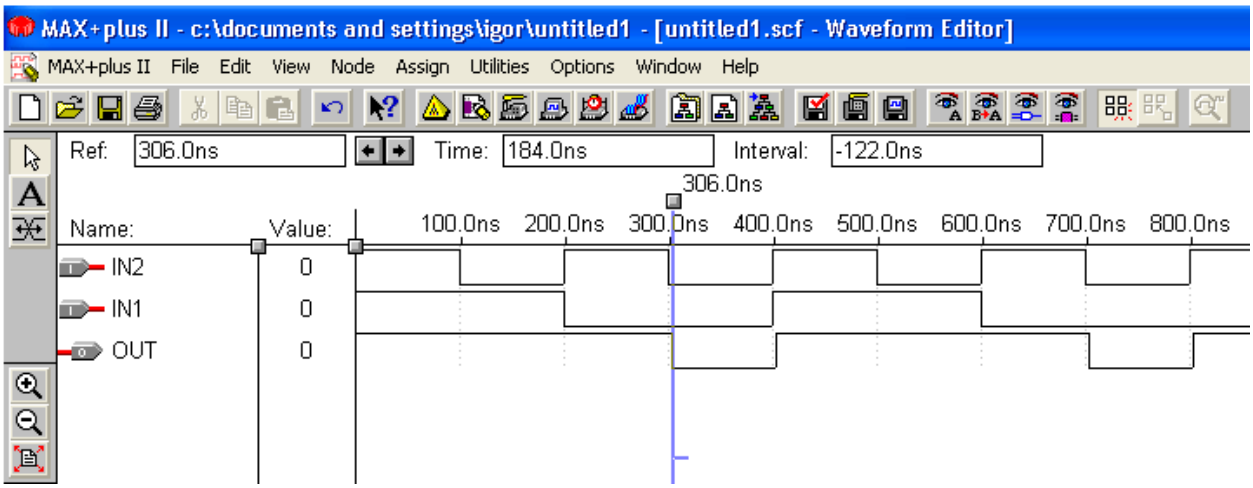


Рис.1.17 — Временные диаграммы входных и выходного сигналов после симуляции

1.4.14. На временной диаграмме обратите внимание на то, как изменилось состояние исходного сигнала ***OUT***. Обратите внимание на то, что выходной сигнал имеет задержку, относительно входных сигналов.

.

1.5. Содержание отчета

1.5.1. Цель работы.
1.5.2. Внешний вид макета.
1.5.3. Скриншот главного меню САПР.
1.5.4. Скриншоты синтезируемых элементов и всех ступеней проектирования.
1.5.5. Выводы

1.6. Контрольные вопросы

1.6.1. Как создать проект?
1.6.2. Что такое редактор графики и символов?
1.6.3. Что такое компилятор?
1.6.4. Что такое симулятор?
1.6.5. Что такое анализатор временных диаграмм?
1.6.6. Что такое программатор?

ЛАБОРАТОРНАЯ РАБОТА № 2

ИЗУЧЕНИЕ БАЗОВЫХ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

2.1. Цель работы:

- Рассмотреть основные законы двоичной алгебры;
- Рассмотреть базовые логические элементы;
- Провести моделирование логических элементов.

2.2. Теоретические сведения

2.2.1. Основные представления о двоичной алгебре и базовых элементах

Все цифровые устройства построены на принципе многократного повторения относительно простых базовых логических схем. Математическим инструментом такого построения служит Булева алгебра, называемая так же алгеброй логики.

В отличие от переменной в обычной алгебре логическая переменная имеет только два значения, которые обычно называются логическим нулем и логической единицей. В качестве обозначений используют «0» и «1» или просто 0 и 1.

Существует три основных операции между логическими переменными, представленные в таблице 2.1.

Таблица 2.1 — Основные логические операции

Операция	Обозначение
Конъюнкция (логическое умножение)	$y = x_1 \wedge x_2 = x_1 \cdot x_2 = x_1 x_2$
Дизъюнкция (логическое сложение)	$y = x_1 \vee x_2 = x_1 + x_2$
Инверсия (логическое отрицание)	$y = \bar{x}$

Для пояснения основных операций алгебры логики на рис. 2.1 представлена их реализация с помощью контактных цепей.

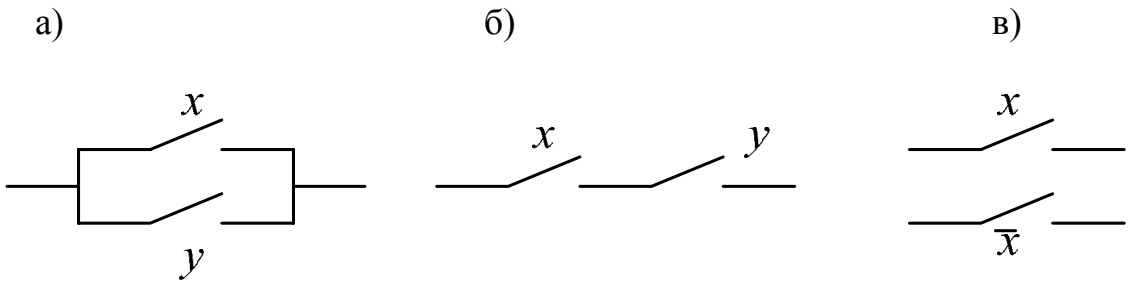


Рис. 2.1 — Реализация базовых логических элементов с помощью контактных цепей

Операция дизъюнкции реализуется параллельным соединением контактов *x* и *y* (рис. 2.1 а), операция конъюнкции — последовательным соединением контактов *x* и *y* (рис. 2.1 б), а операция отрицания — использованием нормально замкнутого контакта (рис. 2.1 в), если считать, что x — нормально разомкнутый контакт.

Применительно к логическим операциям существуют правила и законы булевой алгебры, приведенные ниже.

Коммутативный закон:

$$x_1 x_2 = x_2 x_1 \qquad x_1 + x_2 = x_2 + x_1$$

Ассоциативный закон:

$$x_1(x_2 x_3) = (x_1 x_2) x_3 \qquad x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$$

Дистрибутивный закон:

$$x_1(x_2 + x_3) = x_1 x_2 + x_1 x_3 \qquad x_1 + x_2 x_3 = (x_1 + x_2)(x_1 + x_3)$$

Правило склеивания

$$x_1(x_1 + x_2) = x_1 \qquad x_1 + x_1 x_2 = x_1$$

Правило повторения

$$xx = x \qquad x + x = x$$

Правило отрицания

$$x \cdot \overline{x} = 0 \qquad x + \overline{x} = 1$$

Правило двойного отрицания

$$\overline{(\bar{x})} = x$$

Теорема де Моргана

$$\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2 \qquad \overline{x_1 + x_2} = \bar{x}_1 \cdot \bar{x}_2$$

Операции с 0 и 1

$$x \cdot 1 = x \qquad x + 0 = x$$

$$x \cdot 0 = 0 \qquad x + 1 = 1$$

$$\bar{0} = 1 \qquad \bar{1} = 0$$

С помощью перечисленных законов можно вычислить результаты конъюнкции и дизъюнкции для всех возможных значений переменных x_1 и x_2. Таблица, включающая результаты вычисления по заданному закону называется таблицей соответствия или таблицей истинности.

Таблица 2.2 — Таблица истинности для логического умножения (конъюнкции)

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 2.3 — Таблица истинности для логического сложения (дизъюнкции)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Из таблицы 2.2 следует, что y только тогда равен 1, когда x_1 и x_2 равны 1. На этом основании операция конъюнкции называется также функцией И. При дизъюнкции двух переменных y равен 1 тогда, когда x_1 или x_2 равны 1. Поэтому операцию дизъюнкции называют также функцией ИЛИ. Обе эти функции можно распространить на сколь угодно большое число переменных.

2.2.2. Составление логических функций

Наиболее простым способом задания свойств схемы является таблица истинности. Для реализации схемы требуется найти такую логическую функцию, которая соответствует этой таблице. Далее эту функцию преобразуют в простейшую форму, которую потом реализуют при помощи комбинации базовых логических схем. При условии записи функции в дизъюнктивной нормальной форме выполняется следующая последовательность действий:

1. В таблице истинности выделяют строки, в которых выходная переменная имеет значение 1;
2. Для каждой такой строки составляют конъюнкцию всех входных переменных, причем сомножитель x_i записывают в том случае, если рассматриваемая переменная имеет значение 1, в противном случае записывают $\overline{x_i}$. Т.е. составляется столько произведений, сколько имеется строк с $y=1$;
3. Записывается логическая сумма всех полученных ранее произведений.

Рассмотрим последовательность действий на примере произвольной таблицы истинности.

Таблица 2.4 — Таблица истинности произвольного элемента

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Переменная y принимает значение 1 в 3, 5 и 7 строках. Составим конъюнкцию для этих строк.

$$K_3 = \overline{x_1} x_2 \overline{x_3}$$
$$K_5 = x_1 \overline{x_2} \overline{x_3}$$
$$K_7 = x_1 x_2 \overline{x_3}$$

Результирующую функцию запишем как логическую сумму произведений:

$$y = K_3 + K_5 + K_7 = \overline{x_1} x_2 \overline{x_3} + x_1 \overline{x_2} \overline{x_3} + x_1 x_2 \overline{x_3}$$

Применяя законы двоичной алгебры, приведенные ранее получим конечную запись функции:

$$y = (x_1 + x_2) \cdot \overline{x_3}$$

Полученная функция легко реализуется при помощи базовых логических элементов.

2.3. Порядок выполнения работы

2.3.1. По заданной таблице истинности получить результирующую логическую функцию. Столбец y_N выбирать в соответствии с номером бригады.

x_1	x_2	x_3	y_N	y_N	y_N	y_N	y_N	y_N	y_N	y_N	y_N	y_N
0	0	0	1	0	0	0	0	0	1	0	1	0
0	0	1	0	1	0	1	0	1	0	0	1	0
0	1	0	0	0	1	0	1	0	0	1	1	1
0	1	1	1	0	0	1	0	1	0	1	0	0
1	0	0	0	0	1	0	0	0	1	1	0	1
1	0	1	0	1	0	1	1	0	0	0	0	0
1	1	0	1	0	1	0	0	1	1	0	0	1
1	1	1	0	1	0	0	1	0	0	0	0	0

2.3.2. Составить логическую схему для полученной функции.

2.3.3. Построить данную схему в «MAX +PLUS II», назначив соответственно три входных сигнала и один выходной.

2.3.4. Произвести компиляцию проекта

2.3.5. Провести симуляцию работы схемы при всех сочетаниях входных переменных. Для чего необходимо сформировать три последовательности импульсов входных сигналов типа меандр с периодом последующего сигнала в два раза большим, чем период предыдущего. Это можно сделать вручную, выделяя соответствующие секторы и формируя на заданных временных интервалах лог. 0 или лог. 1. Это можно сделать автоматически, выбрав режим задания тактовых последовательностей для одного или группы узлов (кнопки. «1» или «2» рис.2.2) .

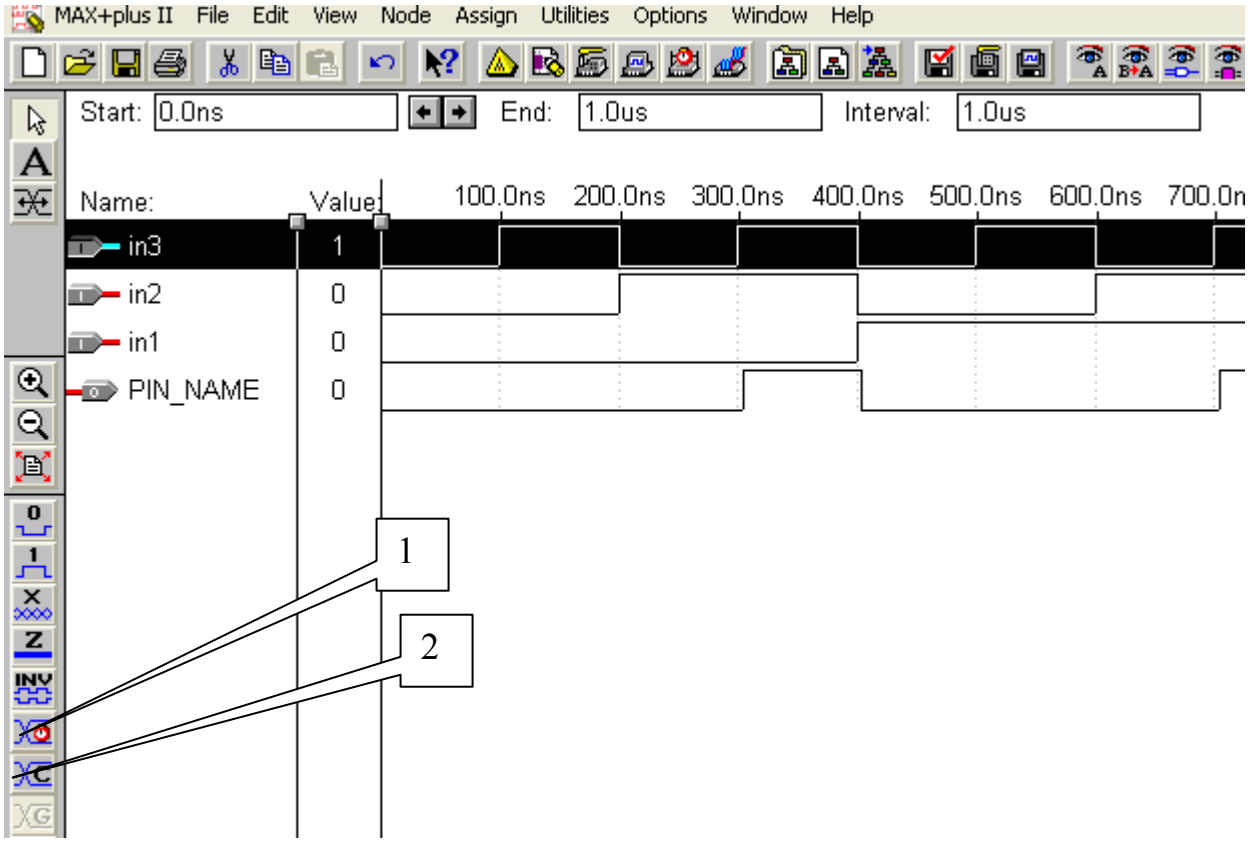


Рис.2.2 — Задание временных последовательностей

Выбирая второй способ задания тактовых импульсов, можно формировать последовательности импульсов с периодами, кратными периоду исходной последовательности (2, 3, 4 и т.д.). Для представления во временной области последовательного перебора всех комбинаций входных переменных, периоды последовательностей

должны быть кратны степени числа 2.

По умолчанию среда разработки *MAX+PLUS* II обеспечивает анализ работы схемы на временном интервале 0…1 *us* с шагом 100 *ns*, что позволяет пронаблюдать всего 10 возможных комбинаций входных переменных или 10 возможных состояний схемы. Для каждого из входных сигналов можно задавать конечное время анализа в отдельности, воспользовавшись меню *File*, опцией *End Time*. По умолчанию *End Time = 1us*. Причем установку времен анализа необходимо начинать с наименьшего. Например, (см. рис.2.3) время изменения сигнала по входу «3» — 2.2 us. (устанавливается первым), по входу «2» — 5.2 *us* (вторым), по входу «1» — 10 us. Для примера четвертая временная диаграмма отображает выходной сигнал схемы 3И при подаче на ее входы описанных временных последовательностей.

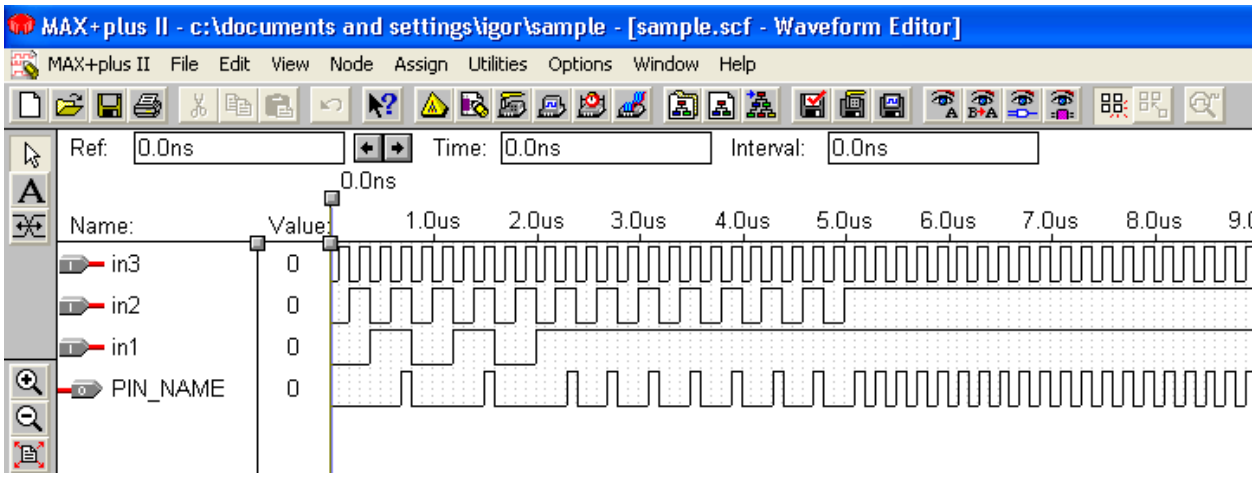


Рис. 2.3 —Временные диаграммы входных сигналов для различных времен анализа

2.4. Содержание отчета

2.4.1. Цель работы.

2.4.2. Заданная таблица истинности.

2.4.3. Составленная формула.

2.4.4. Полученная логическая схема.

2.4.5. Скриншоты **в**ременных диаграмм работы схемы.

2.4.6. Вывод.

2.5. Контрольные вопросы.

2.5.1. Запишите таблицу истинности для элемента 2И.

2.5.2. Запишите таблицу истинности для элемента 2ИЛИ.

2.5.3. Запишите таблицу истинности для элемента 2И-НЕ.

2.5.4. Запишите таблицу истинности для элемента 2ИЛИ-НЕ.

2.5.5. Запишите таблицу истинности для элемента ИСКЛЮЧАЮЩЕЕ ИЛИ.

2.5.6. Запишите таблицу истинности для элемента ИСКЛЮЧАЮЩЕЕ ИЛИ-НЕ.

ЛАБОРАТОРНАЯ РАБОТА №3

ИЗУЧЕНИЕ ДЕШИФРАТОРОВ КОДОВ

3.1. Цель работы:

— Изучить принципы построения дешифраторов на примере десятичного и семисегментного дешифраторов;
— Изучить принципы построения логической схемы по произвольной таблице истинности

3.2. Теоретические сведения

Дешифраторы позволяют преобразовывать одни виды бинарных кодов в другие. Например, преобразовывать позиционный двоичный код в линейный восьмеричный или шестнадцатеричный. Преобразования производятся по правилам, описанным в таблице истинности. Дешифраторы выпускаются в виде отдельных микросхем или используются в составе других микросхем. В настоящее время дешифраторы используются как составная часть других микросхем, таких как мультиплексоры, демультиплексоры, ПЗУ или ОЗУ.

3.2.1. Двоично-десятичный дешифратор

Рассмотрим пример построения дешифратора из двоичного кода в десятичный. Десятичный код обычно отображается одним битом на одну десятичную цифру. В десятичном коде десять цифр поэтому для отображения одного десятичного разряда требуется десять выходов дешифратора. Сигналы с этих выходов можно подать на десятичный индикатор. Таблица истинности такого дешифратора приведена в таблице 1.

Таблица 3.1 — Таблица истинности двоично-десятичного дешифратора

Входы				Выходы									
X8	X4	X2	X1	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Схема, соответствующая данной таблице истинности, приведена на рис.3.1.

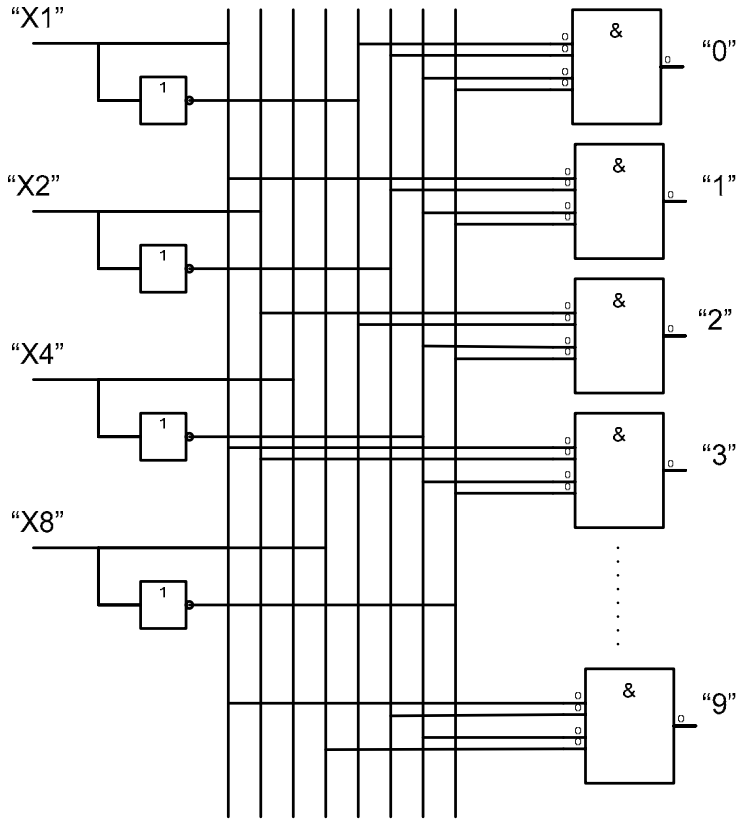


Рис. 3.1 — Схема двоично-десятичного дешифратора

3.2.2. Семисегментный дешифратор

Для отображения десятичных и шестнадцатеричных цифр часто используется семисегментный индикатор. Изображение семисегментного индикатора и название его сегментов приведено на рис. 3.2.

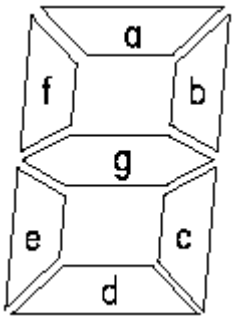


Рис. 3.2 — Семисегментный индикатор

Для отображения на таком индикаторе цифры ноль достаточно зажечь сегменты *a, b, c, d, e,f*. Для отображения цифры 1 зажигают сегменты *b* и *c*. Точно таким же образом можно получить изображения всех остальных десятичных или шестнадцатеричных цифр. Все комбинации таких изображений получили название семисегментного кода. Таблица истинности семисегментного дешифратора приведена ниже.

Таблица 3.2 — Таблица истинности семисегментного дешифратора

Цифра	*X*8	*X*4	*X*2	*X*1	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

3.2.3. Составление схемы семисегментного дешифратора

Первым шагом составления схемы является получение логических функций для каждого из выходов *a*, *b*, *c* ...*g*. При анализе таблицы истинности нетрудно заметить, что число нулей в ней гораздо больше числа единиц. Таким образом, составлять функцию можно для нулевых ячеек таблицы, инвертируя результат. Так, например, для выхода *a* можно записать выражение:

$$a = \overline{\overline{X1} \cdot X2 \cdot \overline{X4} \cdot X8 + \overline{X1} \cdot \overline{X2} \cdot \overline{X4} \cdot X8}$$

Аналогично выполняется построение функций для остальных выходов дешифратора.

По полученной функции можно построить схему дешифратора сегмента *a*.

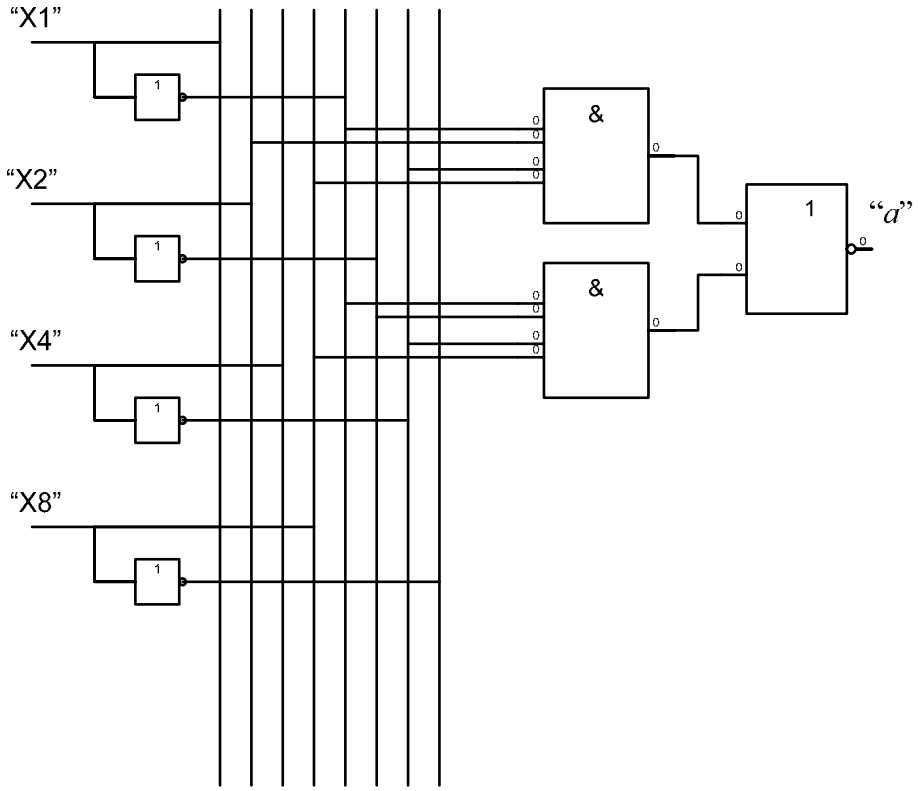


Рис. 3.3 — Принципиальная схема семисегментного индикатора

3.3. Порядок выполнения работы

3.3.1. По таблице истинности составить логические функции всех выходов.

3.3.2. Составить схему семисегментного дешифратора.

3.3.3. Сохранить схему как макрос.

3.3.4. Выполнить симулирование работы схемы, убедиться, что схема составлена верно и работает.

3.3.5. Предъявить результаты моделирования преподавателю. В том случае, если результаты верные, выполнить программирование макета.

3.3.6. Проверить работу дешифратора, изменяя входную комбинацию при помощи тумблеров на макете.

3.4. Содержание отчета

3.4.1. Цель работы.

3.4.2. Таблица истинности семисегментного дешифратора.

3.4.3. Схема семисегментного дешифратора. Скриншот схемы.

3.4.4. Результаты моделирования работы схемы. Скриншот симуляции.

3.4.5. Выводы.

3.5. Контрольные вопросы

3.5.1. Что такое полный дешифратор?

3.5.2. Что такое линейный десятичный дешифратор?

3.5.3. Что такое семисегментный дешифратор?

3.5.4. Как сохранить схему дешифратора как один элемент?

ЛАБОРАТОРНАЯ РАБОТА №4

ИЗУЧЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ

4.1. Цель работы:

— Цель работы: научиться синтезировать асинхронный десятичный счетчик.

— Закрепить навыки создания и анализа работы цифровых схем с помощью САПР «*MAX+PLUS II*»

— Ознакомиться с возможностью замены стандартных цифровых элементов и схем микросхемами программируемой логики.

4.2. Теоретические сведения

4.2.1. Посторенние счетчика

Счетчиком называется последовательностная схема циклически переходящая из одного устойчивого состояния в другое под действием входных сигналов. При этом можно получить различный коэффициент пересчета (*модуль счета*) Кс - количество входных переключающих импульсов, которое нужно подать на вход счетчика, чтобы он вернулся в исходное состояние.

Простейшим счетчиком является триггер. В счетном режиме на его вход нужно подать 2 счетных импульса, чтобы триггер возвратился в исходное состояние.

Для того, чтобы получить произвольный модуль счета *Кс* необходимо

$$l = \log_2 K_C \text{ триггеров}$$

Если $K_c = 2^m$, где *m* — целое, то такой счетчик называют двоичным.

Если $K_c \neq 2^m$, то для организации такого счетчика потребуется *n* триггеров, где *n* — ближайшее целое большее *l*.

Состояние счетчика определяется состоянием его триггеров

$$Q_1, Q_2, Q_3, \quad Q_n.$$

Правило работы счетчика, т.е. порядок изменения его состояний задают в ви-

де соответствующей таблицы. В счетчиках с естественным порядком счета каждый входной переключающий сигнал приводит к изменению числа, зафиксированного в счетчиках на единицу. В зависимости от знака этого изменения счетчики различают на суммирующие и вычитающие.

В таблице 4.1 заданы переходы трехразрядного двоичного счетчика из одного состояния в другое при подаче на его вход *N* запускающих импульсов.

Из таблицы видно, что при поступлении на вход счетчика 8 импульсов, последний возвращается в исходное состояние, т.е.

$$K_c = 2^3 = 8,$$

где 3 — число разрядов (триггеров) счетчика.

Если рассматривать таблицу сверху вниз, то каждый счетный импульс увеличивает содержимое счетчика на единицу — реализуется суммирующий режим работы счетчика. Если эту же таблицу рассматривать снизу вверх — получаем вычитающий режим работы счетчика.

Таблица 4.1 — Таблица переходов двоичного счетчика

N	Q3	Q2	Q1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
9	0	0	1
10	0	1	0
11	0	1	1
и т.д.			

Для пояснения счетного режима работы счетчика на рис.4.1 приведена временная диаграмма работы суммирующего двоичного счетчика с модулем счета равным 8.

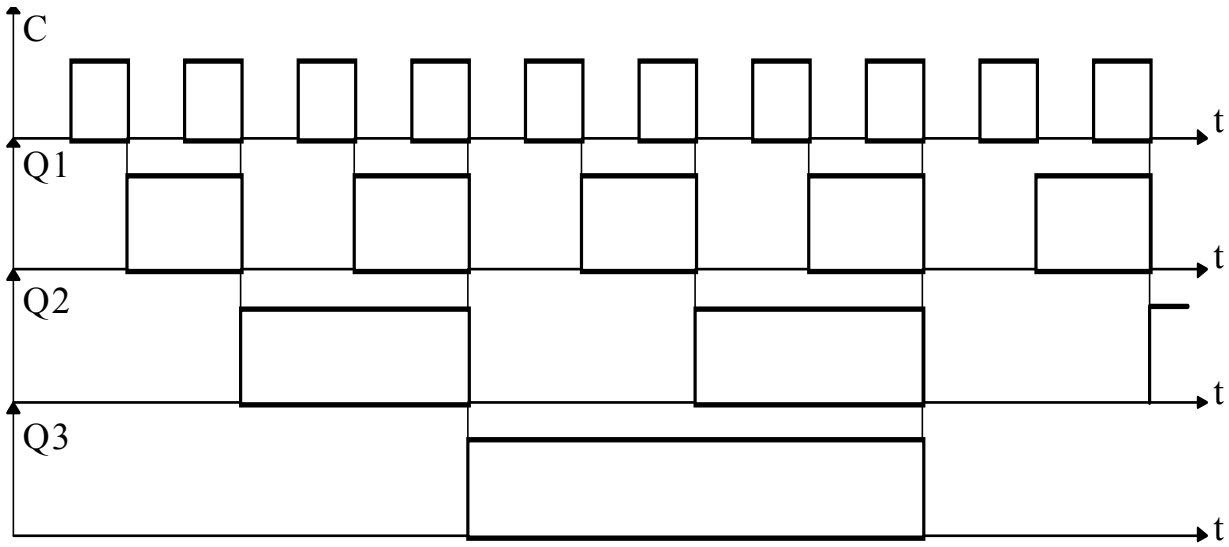


Рис.4.1 — Временная диаграмма работы двоичного счетчика

Для организации модуля счета асинхронного счетчика не равного степени 2, например 5, необходимо организовать асинхронный сброс всех триггеров счетчика в исходное нулевое состояние при поступлении пятого по счету импульса синхронизации. Пятый по счету импульс синхронизации согласно временной диаграмме, представленной на рис. 4.1 устанавливает первый триггер в состояние “1”, второй триггер к этому моменту времени находился в состоянии “0”, третий - “1”. Таким образом, объединяя выходы первого и третьего триггеров логикой “И”, можно синтезировать сигнал, свидетельствующий о том, что счетчик находится в состоянии “101” (5) и по этому сигналу организовать асинхронный сброс всех триггеров счетчика в исходное нулевое состояние.

При организации вычитающего счетчика исходным состоянием будет такое, когда все триггеры счетчика находятся в состоянии “1”.

Принципиальная схема асинхронного суммирующего счетчика по модулю 5 приведена на рис. 4.2, а временная диаграмма, поясняющая его работу на рис. 4.3.

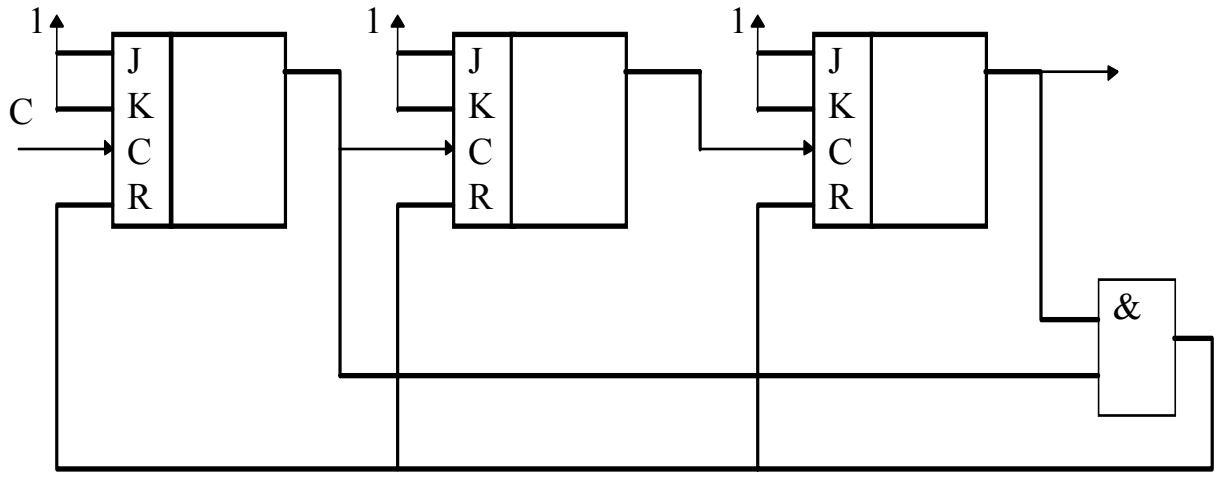


Рис. 4.2 — Принципиальная схема произвольного счетчика

Особенностью работы счетчика является то, что на выходе схемы “И” формируются очень короткие импульсы, зарегистрировать которые с помощью обычных средств наблюдения за процессом работы, например, осциллографами, как правило не представляется возможным, так как длительность этих импульсов составляет $2t_d$, где t_d — время задержки переключения триггеров и логической схемы “И”, определяемой технологией изготовления этих элементов. Как правило, это время составляет от единиц до сотен наносекунд, и зарегистрировать его достаточно сложно.

Таким образом, модуль счета асинхронного счетчика может быть спроектирован произвольным, для этого достаточно объединить логикой 2- , 3- и более “И” соответствующие выходы триггеров счетчика, синтезируя тем самым сигнал асинхронного сброса всех триггеров счетчика в исходное нулевое состояние.

На практике иногда представляет интерес не сброс всех триггеров счетчика, а их предустановка в состояние “1”, либо промежуточный вариант, когда часть триггеров сбрасывается в “0”, а часть устанавливается в “1”. В каждом конкретном случае необходимо выбирать тот вариант, который дает наименьшее число элементов и их связей.

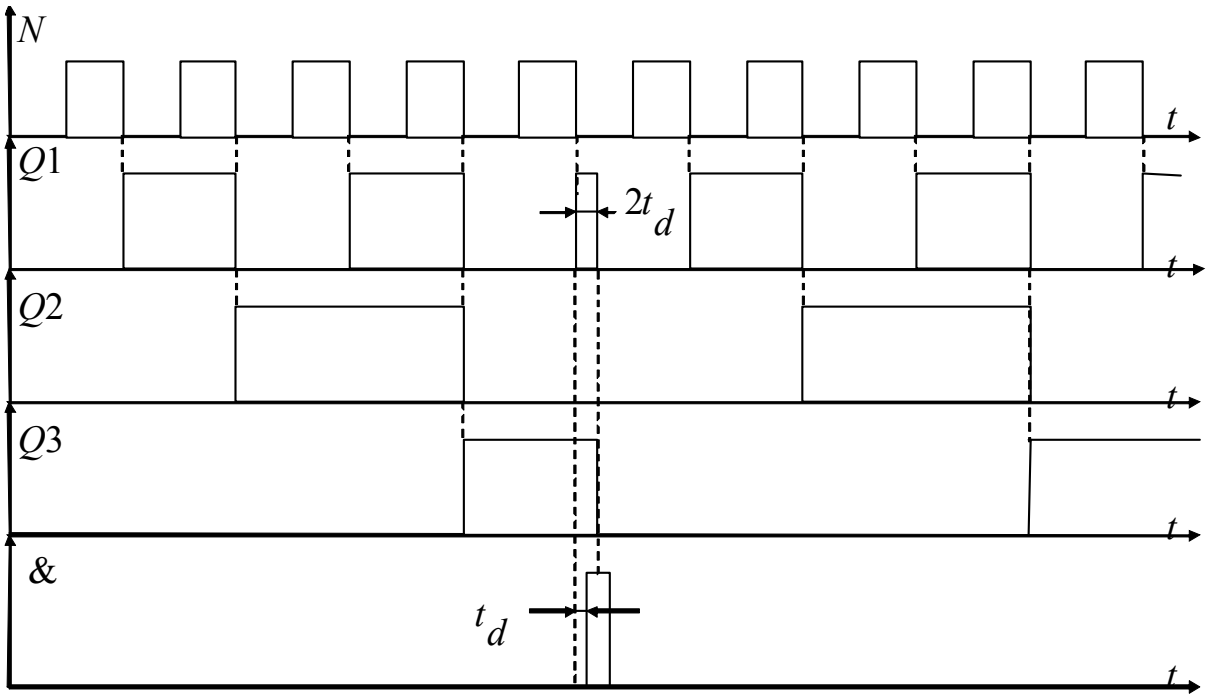


Рис. 4.3 — Временная диаграмма работы произвольного счетчика

4.2.2. Реализация счетчика в системе «*MAX+PLUS II*»

Для иллюстрации возможностей программируемой логики рассмотрим создание на базе ПЛИС простейшего счетчика количества импульсов.

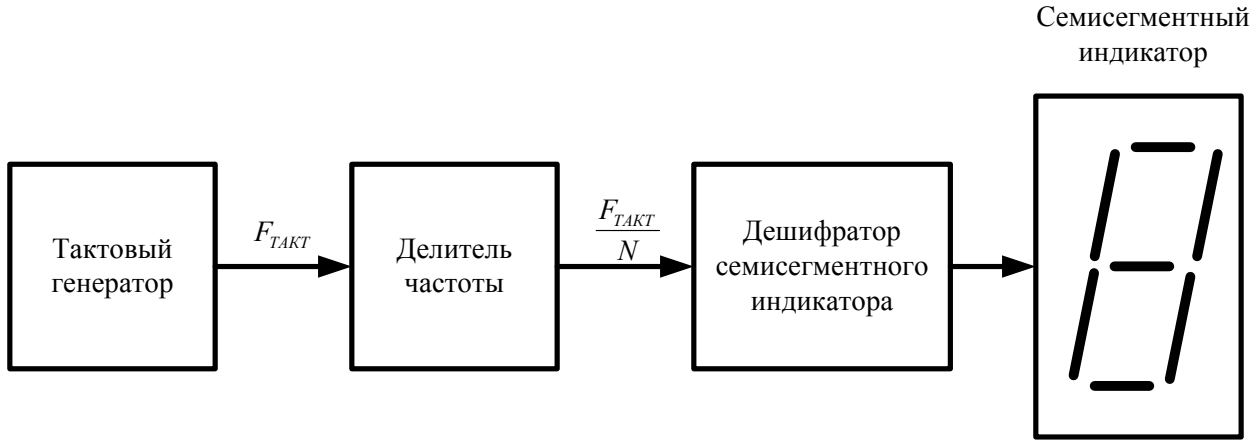


Рис. 4.4 — Структурная схема простейшего счетчика количества импульсов

В схеме, показанной на рис. 4.4 используется тактовый генератор, который формирует сигнал с частотой $F_{\text{ТАКТ}}$. Этот сигнал поступает на вход делителя частоты. Частота выходного сигнала делителя частоты меньше частоты входного сигнала в N раз. Коэффициент деления N делителя частоты определяется его внутренней организацией.

С выхода делителя частоты сигнал с частотой $F_{\text{ТАКТ}}/N$ поступает на вход

дешифратора семисегментного индикатора, который преобразовывает входной сигнал в сигнал управления семисегментным индикатором.

В итоге, при поступления на вход схемы от тактового генератора *N* импульсов, показания индикатора изменятся на единицу.

Для выполнения такой схемы на обычных цифровых микросхемах потребовалось бы не менее двух корпусов (при увеличении *N* и количества индикаторов это число быстро возрастает). При использовании микросхем программируемой логики ситуация упрощается.

Рассмотрим реализацию счетчика с коэффициентом деления 1024. Это число может быть записано как 2^{10}. Такой коэффициент деления могут обеспечить 10 последовательно соединенных *D*-триггеров. На рис. 4.5 показаны символы наиболее распространенных триггеров, а на рис. 4.6 показана принципиальная схема делителя на 1024, выполненная на *D*-триггерах.

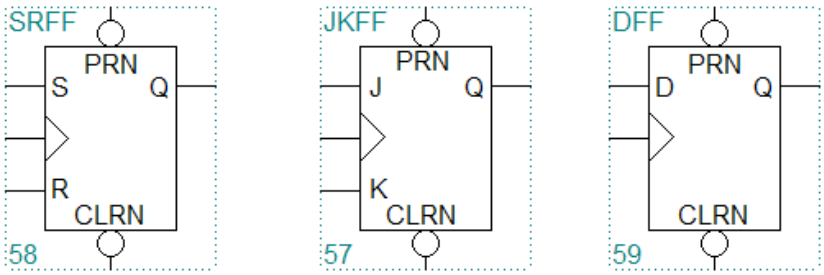


Рис. 4.5 — Элементы САПР для *RS*, *JK* и *D*-триггеров

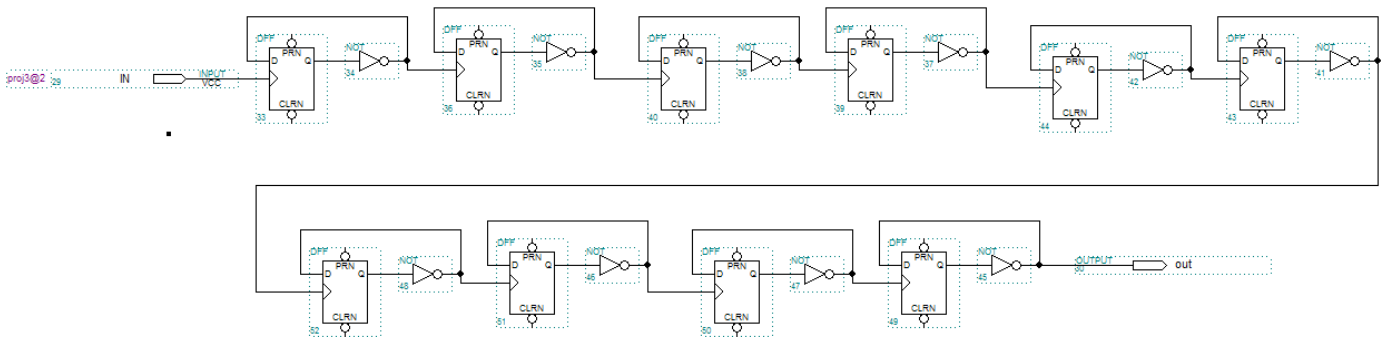


Рис. 4.6 — Предварительный делитель частоты на 1024

Для анализа работы схемы счетчика понадобится дешифратор, который был создан Вами на предыдущем занятии и дополнительный четырехразрядный счетчик импульсов, осуществляющий дополнительное деление полученного значения частоты в 2, 4, 8 и 16 раз. При этом, если величина входной частоты будет составлять порядка одного килогерца, показания индикатора будут обновляться примерно через

секунду (то есть в 1024 раз медленнее). В итоге, проектируемый счетчик будет выглядеть так, как показано на рис. 4.7.

4.3. Порядок выполнения работы

4.3.1. Синтезировать схему асинхронного двоичного счетчика с модулем счета, равным 1024. Сохранить сему как один элемент.

4.3.2. К выходу схемы (выход последнего триггера счетчика) подключить один из светодиодов.

4.3.3. Ко входу схемы подключить тактовый генератор. Тактовая частота поступает на вход *CLOCK* (43 вывод ПЛИС, см. ЛР1). Величина тактовой частоты составляет порядка 1 кГц. В качестве генератора использован интегральный таймер.

4.3.4. Скомпилировать схему и запрограммировать ПЛИС.

4.3.5. Пронаблюдать работу счетчика, оценить период мигания светодиода.

4.3.6. Синтезировать схему счетчика с модулем счета, определяемым по индивидуальному заданию. Индивидуальные задания приведены в таблице 4.2. Сохранить схему как один элемент.

Таблица 4.2 — Индивидуальные задания

№ бригады	1	2	3	4	5	6	7	8	9	10
Модуль счета	5	6	7	9	10	11	12	13	14	15

4.3.7. Провести симуляцию работы схемы счетчика, разработанного по индивидуальному заданию.

4.3.8. К выходам триггеров счетчика, подключить синтезированный ранее семисегментный дешифратор.

4.3.9. На вход счетчика подать сформированные секундные импульсы, добавив в схему синтезированный ранее двоичный счетчик с модулем счета 1024.

4.3.10. Скомпилировать схему устройства и запрограммировать ПЛИС.

4.3.11. Пронаблюдать работу счетчика на семисегментном индикаторе.

4.4. Содержание отчета

4.4.1. Цель работы.

4.4.2. Скриншот схемы предварительного делителя на 1024.

4.4.3. Скриншот схемы счетчика импульсов по индивидуальному заданию.

4.4.4. Результаты симулирования работы схемы. Скриншот симуляции.

4.4.5. Скриншот всего проекта.

4.4.5. Выводы.

4.5. Контрольные вопросы

4.5.1. Что такое последовательностная схема?

4.5.2. Что такое двоичный счетчик импульсов?

4.5.3. Что такое счетчик импульсов с произвольным модулем счета?

4.5.4. Что такое асинхронный счетчик импульсов?

4.5.5. Что такое синхронный счетчик импульсов?

ЛАБОРАТОРНАЯ РАБОТА №5

СИНТЕЗ ЛОГИЧЕСКИХ СХЕМ

5.1. Цель работы:

— Закрепить навыки синтеза схем на основе ПЛИС и анализа работы цифровых схем с помощью САПР «*MAX+PLUS II*».

5.2. Теоретические сведения

Теоретические сведения включают повторение теоретических сведений лабораторных работ №1…№4.

5.3. Индивидуальные задания

5.3.1. Вариант 1

С использованием полученных ранее макросов синтезировать схему, обеспечивающую мерцание светодиода LED 1 с частотой 1 Гц и подсчет количества мерцаний с выводом десятичной цифры на индикаторах HL1, HL2. Обнуление счетчика должно происходить по нажатию клавиши SA1.

5.3.2. Вариант 2

С использованием полученных ранее макросов синтезировать схему, обеспечивающую мерцание светодиода LED 1 с частотой 1 Гц и подсчет количества мерцаний с выводом цифры в двоичной системе на индикаторах HL1, HL2. Обнуление счетчика должно происходить по нажатию клавиши SA1.

5.3.3. Вариант 3

С использованием полученных ранее макросов синтезировать схему, обеспечивающую мерцание светодиода LED 1 с частотой 1 Гц и подсчет количества мерцаний с выводом цифры в восьмеричной системе на индикаторах HL1, HL2. Обну-

ление счетчика должно происходить по нажатию клавиши SA1.

5.3.4. Вариант 4

С использованием полученных ранее макросов синтезировать схему, обеспечивающую поочередное мерцание светодиодов, начиная от LED 1 и заканчивая LED 8, с частотой 1 Гц и подсчет количества мерцаний LED 3 с выводом цифры в десятичной системе на индикаторах HL1, HL2. Обнуление счетчика должно происходить по нажатию клавиши SA1.

5.3.5. Вариант 5

С использованием полученных ранее макросов синтезировать схему, обеспечивающую мерцание светодиодов, начиная от LED 1 и заканчивая LED 8, частотой 0,5 Гц и подсчет количества мерцаний LED с выводом цифры в двоичной системе на индикаторах HL1, HL2. Обнуление счетчика должно происходить после 30 мерцаний.

5.2.6. Вариант 6

С использованием полученных ранее макросов синтезировать схему, обеспечивающую подсчет нажатий на кнопку SA1 с индикацией количества нажатий на светодиодном индикаторе HL1, HL2 и индикацией двоичного кода при помощи светодиодов LED1…LED8 согласно весовым коэффициентам.

5.3.7. Вариант 7

С использованием полученных ранее макросов синтезировать схему, обеспечивающую подсчет нажатий на кнопку SA1 с индикацией количества нажатий на светодиодном индикаторе HL1, HL2 и сбросом счетчика по нажатию кнопки SA2.

5.3.8. Вариант 8

С использованием полученных ранее макросов синтезировать схему, работу макета в режиме таймера с периодом 60 секунд. Запуск таймера по нажатию на кнопку SA1, сброс по нажатию на кнопку SA2. По достижению цифры 60 должны быть засвечены все светодиоды от LED1 до LED 8.

5.4. Порядок выполнения работы

5.4.1. Синтезировать проектируемую схему по индивидуальному заданию. Сохранить сему как один элемент.

5.4.2. К выходам схемы подключить светодиодные индикаторы по индивидуальному заданию.

5.4.3. Ко входу схемы подключить тактовый генератор. Тактовая частота поступает на вход *CLOCK* (43 вывод ПЛИС, см. ЛР1). Величина тактовой частоты составляет порядка 1 кГц. В качестве генератора использован интегральный таймер.

5.4.4. Ко входам схемы подключить кнопки по индивидуальному заданию.

5.4.5. Скомпилировать схему и запрограммировать ПЛИС.

5.4.6. Пронаблюдать работу схемы.

5.4.7. Провести симуляцию работы схемы,.

5.5. Содержание отчета

5.5.1. Цель работы.

5.5.2. Индивидуальное задание.

5.5.3. Скриншот проектируемой схемы.

5.5.4. Результаты симулирования работы схемы. Скриншот симуляции.

5.5.5. Выводы.

5.6. Контрольные вопросы

В качестве контрольных вопросов используются контрольные вопросы всех предыдущих лабораторных работ.

Список ссылок

1. ?