Министерство образования Российской Федерации

Владимирский государственный университет

А.Н.Волков, М.В.Руфицкий

ПРОЕКТИРОВАНИЕ ЭЛЕКТРОННЫХ СРЕДСТВ НА ОСНОВЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

Классификация, технология изготовления, маршрут проектирования

Рекомендовано учебно-методическим объединением по электронике Российской Федерации в качестве учебного пособия для студентов высших учебных заведений, обучающихся по специальностям «Проектирование и технология электронно-вычислительных средств», «Проектирование и технология радиоэлектронных средств»

Владимир 2002

УДК 621.375(03)

В67

Рецензенты

Доктор технических наук, профессор Российского государственного технологического университета им. К.Э. Циолковского

С.А. Пескова

Доктор технических наук, профессор кафедры компьютерных систем автоматизации производства Московского государственного технического университета им. Н.Э Баумана

В.В. Емельянов

Печатается по решению редакционно-издательского совета

Владимирского государственного университета

В67 **Волков А.Н., Руфицкий М.В.** Проектирование электронных средств на основе программируемых логических интегральных схем. Классификация, технология изготовления, маршрут проектирования: Учеб. пособие; Владим. гос. ун-т; Владимир, 2002. 112 с.

ISBN 5-89368-302-1

Представлено подробное описание конструктивно-технологических особенностей ПЛИС. Рассмотрен маршрут проектирования ПЛИС и особенности современных САПР ПЛИС. Методология проектирования устройств на ПЛИС представлена в виде курса лабораторных работ, охватывающих весь цикл проектирования ПЛИС с выходом на практическое использование.

Предназначено для студентов, обучающихся по специальностям: 20.08.00 - проектирование и технология радиоэлектронных средств и 22.05.00 - проектирование и технология электронно-вычислительных средств.

Табл. 25. Ил. 56. Библиогр.: 16 назв.

ISBN 5-89368-302-1

Оглавление

ВВЕДЕНИЕ ... 3
1. КЛАССИФИКАЦИЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ (ПЛИС) ... 6
1.1. Классификация ПЛИС по степени интеграции ... 6
1.2. Архитектура функционального преобразователя ПЛИС ... 7
1.3. Организация внутренней структуры ПЛИС ... 12
1.4. Наличие внутренней RAM-памяти ... 15
2. ТЕХНОЛОГИЯ ИЗГОТОВЛЕНИЯ КОНФИГУРАЦИОННЫХ ЭЛЕМЕНТОВ ПЛИС ... 16
2.1. Конфигурационный элемент EPROM ... 17
2.2. Конфигурационный элемент EEPROM ... 19
2.3. Конфигурационный элемент FLASH ... 21
2.4. Конфигурационный элемент SRAM ... 23
2.5. Конфигурационный элемент ANTIFUSE ... 23
3. ОБЛАСТИ ПРИМЕНЕНИЯ ПЛИС ... 25
3.1. Достоинства и недостатки ПЛИС ... 26
3.2. Перспективные направления развития ПЛИС ... 26
3.3. Обзор семейств ПЛИС фирмы Altera ... 29
4. ЛАБОРАТОРНЫЙ ПРАКТИКУМ ... 35
4.1. Лабораторная работа №1. Графический редактор системы MAX PLUS II ... 35
4.2. Лабораторная работа №2. Работа с компилятором системы MAX PLUS II ... 49
4.3. Лабораторная работа №3. Логический синтез проекта на ПЛИС ... 60
4.4. Лабораторная работа №4. Моделирование и временной анализ проекта на ПЛИС ... 68
4.5. Лабораторная работа №5. Программирование ПЛИС ... 90
Приложение 1. Структура обозначений ПЛИС фирмы Altera ... 101
Приложение 2. Типы корпусов ПЛИС ... 102
Приложение 3. Примеры корпусов ... 103
Библиографический список ... 111

ВВЕДЕНИЕ

Современный этап развития средств электронной и вычислительной техники характеризуется двумя противоречивыми тенденциями:

- с одной стороны, увеличивается их сложность и ужесточаются требования, предъявляемые потребителями к быстродействию, надёжности, энергопотреблению, стоимости;
- с другой, их жизненный цикл существенно сокращается.

Особое значение в этом случае приобретает время выхода на рынок нового изделия. Следовательно, сроки, установленные на проведение разработки и макетирования, сокращаются, а требования, предъявляемые к качеству новых изделий, становятся всё более жёсткими.

В табл.1 показаны основные этапы развития конструктивно-технологических признаков изделий электронной техники [13]. Если развитие первых поколений техники происходило за счет новых технологий изготовления, соотношения числа транзисторов на кристалле, быстродействия, то в последних поколениях (IV – V) резервы совершенствования технологий практически исчерпаны: любые улучшения достигаются более дорогой ценой. Следовательно, развитие электронной техники будет происходить, в основном, за счет совершенствования систем автоматизированного проектирования и широкого применения специализированных БИС.

Проектирование специализированных БИС позволяет выполнять функции, которые не реализуются в стандартных ИС, улучшать характеристики схем, снижать габаритные размеры, массу, мощность, повышать надежность электронных средств (ЭС). Основу специализированных БИС составили программируемые логические интегральные схемы (ПЛИС). Одним из основных преимуществ ПЛИС

Таблица 1

Классификация конструктивно-технологических признаков изделий электронной техники по поколениям

Признаки	II поколение 1960-1975 г.	III поколение 1975-1985 г.	IV поколение 1985-1995 г.	V поколение 1995-2005 г.	VI поколение 2005-2015 г.
Конструктивные (степень интеграции элем./корп.)	10^0 э/к электровакуумные приборы	Полупроводниковые триоды, диоды, варикапы; ИС (10^2-10^3 э/к)	БИС (10^3-10^4 э/к) Полузаказные БИС (10^4 э/к)	БИС (10^5-10^6 э/к) ПЛИС ($5\cdot10^5$-$2\cdot10^6$ э/к)	СБИС (10^9 э/к)
Технологические (степень автоматизации процессов проектирования, производства, контроля)	Проектирование ручное. Производство ручное. Контроль ручной	Автоматизированное проектирование печатных плат, частично автоматиз. производство, контроль ручной	Автоматизир. проектирование печатных плат, программирование в машинных кодах, проектирование полузаказных БИС, элементы автоматиизированного контроля	Системы проектирования на языках высокого уровня (Cadence, HPEEsof, Mentor Grafic), автоматизированный контроль, элементы автоматизации производства	Обучающиеся системы проектирования. Автоматизированное производство и контроль

является возможность синтеза различных устройств без изменения различных устройств без изменения технологического базиса, а также значительное сокращение сроков проектирования.

Первые ПЛИС в их простейшем варианте (PAL/GAL) появились в конце 70-х годов и являлись вспомогательной элементной базой для создания единичных и малосерийных комбинационных и последовательностных автоматов сложностью до нескольких десятков эквивалентных вентилей 2И-НЕ.

С середины 80-х началась новая эра в развитии ПЛИС. В этот период были основаны три ведущие корпорации – основные разработчики ПЛИС. В июне 1983 г. основана фирма Altera Corporation (101 Innovation Drive, san Jose, CA 95134, USA,www.altera.com), в феврале 1984 – компания Xilinx, Inc. (2100 Logic Drive, San Jose, CA 95123-3400, USA,www.xilinx.com), в 1985 – Actel Corporation (955 East Arques Avenue, Sunnyvale, CA 94086-4533, USA,www.actel.com). Эти три компании занимают до 80 % всего рынка ПЛИС и являются основными разработчиками идеологии их применения. С момента своего основания эти и ряд других компаний (Atmel, Lucent Technologies, LSI Logic, Triscend и др.) активно разрабатывают новые классы и семейства ПЛИС, отличающиеся наличием новых функций. Обозначилась тенденция специализации рынка, когда та или иная компания-разработчик является лидером по одному из направлений.

В последние годы рынок ПЛИС значительно расширился с появлением новых архитектур и семейств микросхем, что накладывает дополнительные обязанности на разработчика аппаратуры: из всего многообразия архитектур и семейств инженер должен выбрать лучший кристалл для своего проекта. Корректный выбор повлечёт за собой успех

на рынке, и наоборот, неудачный выбор приведёт к неоправданным экономическим потерям и последующим переработкам проекта.

1. КЛАССИФИКАЦИЯ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ (ПЛИС)

Разнообразие существующих типов ПЛИС не позволяет выбрать единый критерий для их всеобъемлющей классификации. Поэтому целесообразно выделить набор классификационных критериев, обеспечивающих возможность систематизации информации о характеристиках и особенностях ПЛИС: К основным критериям классификации ПЛИС следует отнести:

- степень интеграции;
- архитектура простейшего функционального преобразователя;
- организация внутренней структуры и структуры матрицы соединений;
- наличие внутренней RAM-памяти;
- технология изготовления программируемых элементов.

Рассмотрим каждый из критериев классификации более подробно.

1.1. Классификация ПЛИС по степени интеграции

Степень интеграции ПЛИС характеризуется логической ёмкостью. Логическая ёмкость измеряется числом логических вентилей и определяет возможность ПЛИС обеспечить реализацию цифрового устройства заданной сложности. Как правило, в качестве базового логического вентиля принимают элемент 2И-НЕ.

В соответствии с выбранным критерием ПЛИС подразделяются на три подгруппы:

- низкой степени интеграции (лог.ёмк. до 1500 ЛВ);

семейства XC3000, XC7000 (ф.Xilinx); семейства ATF, ATV, ATL (ф.Atmel); семейства Classic, MAX5000, MAX7000E/S (ф.Altera); семейство ACT1 (ф.Actel);

- средней степени интеграции (лог.ёмк. от 1500 до 15000 ЛВ);

семейства XC5000, XC9000 (ф.Xilinx); семейство AT6000 (ф.Atmel); семейства Classic, MAX9000, FLEX8000 (ф.Altera);
семейства ACT2, ACT3 (ф.Actel);

- высокой степени интеграции (лог.ёмк. от 15000 до 150000 ЛВ);

семейства XC4000, XC6000, Spartan, VIRTEX (ф.Xilinx); семейство FLEX10K, APEX (ф.Altera).

1.2. Архитектура функционального преобразователя ПЛИС

В настоящее время существует множество архитектур простейших функциональных преобразователей ПЛИС.

Первыми в логическом проектировании цифровых устройств получили применение БИС постоянных запоминающих устройств:: программируемых (ППЗУ,

PROM), репрограммируемых (РПЗУ, EPROM) и электрически перепрограммируемых (ЭППЗУ, EEPROM). ПЗУ имеет структуру универсального логического преобразователя, т.е. функционально является композицией двух блоков (рис.1): дешифратора (D) и шифратора (S).

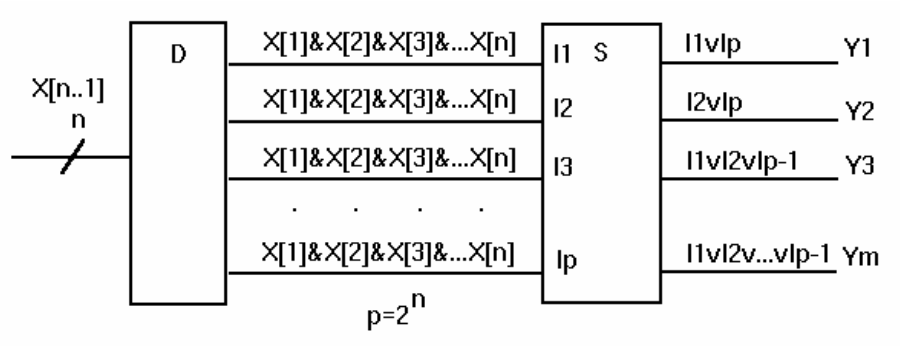


Рис.1. Структура ПЗУ

Дешифратор, имеющий фиксированную структуру, порождает полный набор термов от n входных переменных. Шифратор, реализованный на базе массива хранения данных ПЗУ, является программируемым и обеспечивает формирование m независимых логических функций от n переменных, представленных в совершенной дизъюнктивной нормальной форме. Недостаток такой организации - избыточность представления функции в СДНФ и связанная с ней степенная зависимость объёма накопителя ПЗУ от числа аргументов. Так, для реализации функции от 32 переменных потребуется массив объёмом 2^{32}=4 Гбит.

СБИС программируемой логики следующего поколения - программируемые логические матрицы ПЛМ (Programmable Logic Array - PLA), позволяющие реализовать логические функции, представленные в произвольной (сокращенной, тупиковой, минимальной) дизъюнктивной нормальной форме (ДНФ). ПЛМ так же, как и ПЗУ, имеет структуру универсального логического преобразователя (рис.2). Причём, и дешифратор (матрица ”И”), и шифратор (матрица “ИЛИ”) являются программируемыми. Для обеспечения возможности реализации не только

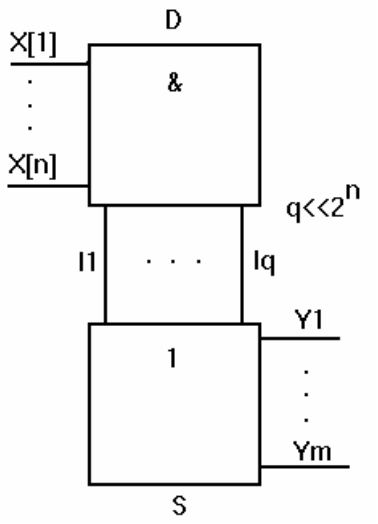


Рис.2. Структура ПЛМ

комбинационных, но и последовательностных схем, на выходы шифратора были добавлены триггеры. Такие БИС получили название программируемых логических контроллеров (ПЛК).

Поскольку у логических функций, представленных в ДНФ, редко бывают общие термы, то матрица “ИЛИ” оказывается разреженной, а занятая ей часть кристалла используется не полностью. Таким образом, недостаток ПЛМ и ПЛК - неэффективное использование матрицы “ИЛИ”. Следующим этапом развития ПЛИС явилось появление БИС матричной логики (ПМЛ), также имеющих структуру универсального логического преобразователя (рис.3): дешифратор (D) - шифратор (S). При этом дешифратор (матрица “И”) у ПМЛ - программируемый, а шифратор (матрица “ИЛИ”) имеет фиксированную структуру. Простейшие БИС ПМЛ за рубежом получили название Programmable Array Logic (PAL), а ПМЛ с регистрами на выходе - Generic Array Logic (GAL).

Архитектура БИС ПМЛ, в англоязычной литературе получившая название Sum-of-Products, оказалась весьма эффективной. Поэтому она была положена в основу организации простейших функциональных преобразователей СБИС новых поколений - программируемых логических устройств (ПЛУ), англоязычное название - Programmable Logic Devices (PLD).

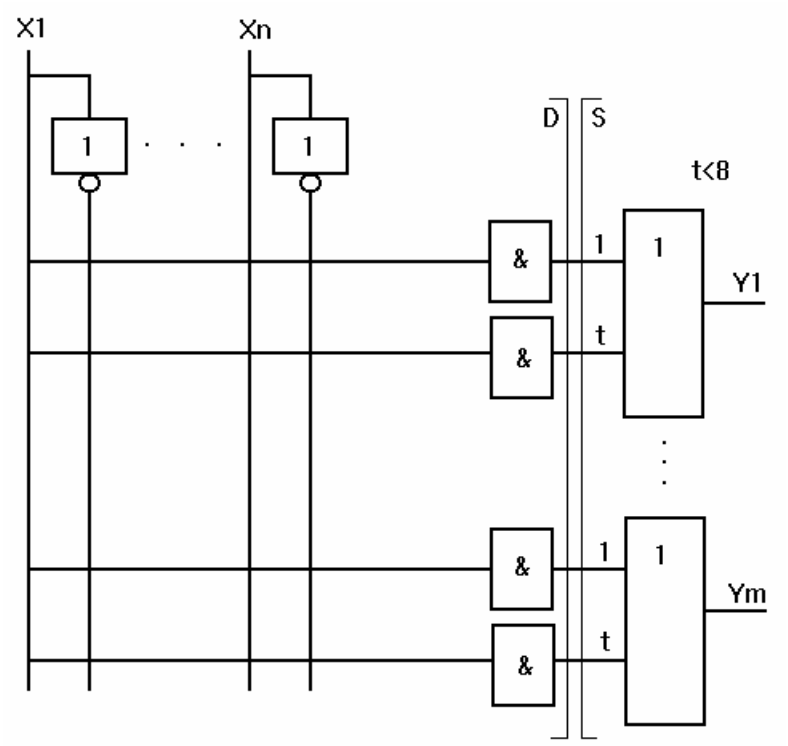


Рис.3. Структура ПМЛ

В настоящее время получили развитие и другие архитектуры простейших функциональных преобразователей.

Одна из них, табличная архитектура, основана на использовании для формирования логических функций таблиц перекодировок (Look-up-table). В общем случае таблица перекодировки выполняет те же функции, что и перепрограммируемые ПЗУ. Обобщённая структура простейшего функционального преобразователя, основанного на этой архитектуре, приведена на рис. 4.

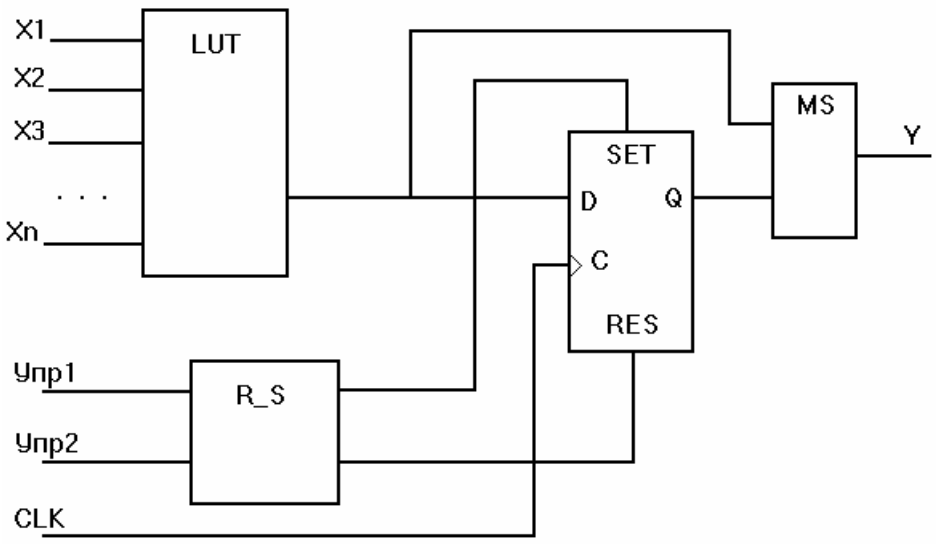
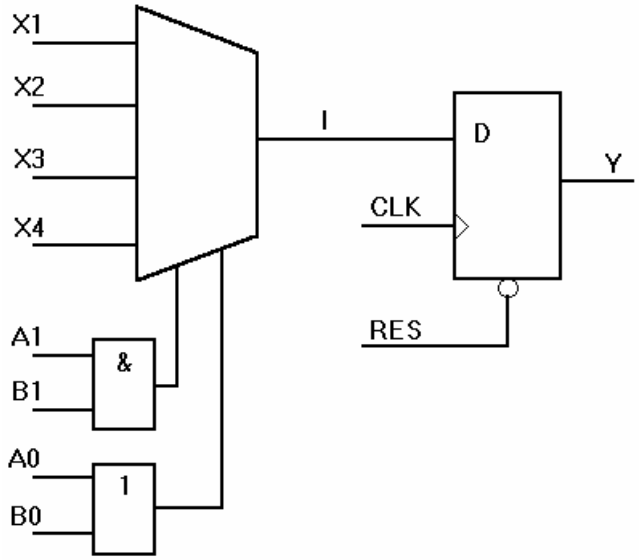


Рис.4. Структура Look-Up-Table

В его состав входят: n-входовая таблица перекодировки (LUT); синхронный триггер с D входом и установками SET и RES; логическая схема управления асинхронными установкой/сбросом триггера (R_S); программируемый мультиплексор выбора источника выходного сигнала (MS). Таблица перекодировок с n входами представляет собой одноразрядное запоминающее устройство объёмом 2^n бит, позволяющее реализовать любую логическую функцию от n переменных.

Таким образом, в рамках простейшего функционального преобразователя, как и в исторически первых СБИС программируемой логики - ПЗУ, для формирования логической функции используется фиксированная матрица “И” и программируемая матрица “ИЛИ”. Однако, в отличие от ПЗУ, число входов n в таблице перекодировок невелико, а число простейших функциональных преобразователей, размещенных в СБИС, наоборот - весьма значительно и может достигать нескольких тысяч, что позволяет осуществлять иерархическую реализацию сложных логических функций, и, тем самым, устранить известный недостаток табличной реализации, связанный со степенной зависимостью объёма ЗУ от числа аргументов функции. Так, для реализации функции от 32 аргументов потребуется всего 11 четырёхвходовых таблиц перекодировок, а не ЗУ объёмом 4 Гбит.

Другой тип архитектуры (Simple Logic Cell) простейшего функционального преобразователя основан на использовании комбинационных схем, обеспечивающих реализацию того или иного, минимального или неминимального базиса. На рис.5 приведена структура простейшего функционального преобразователя СБИС семейства ACT2 фирмы Actel.



$I=\overline{S1}\&\overline{S0}\&X1+\overline{S1}\&S0\&X2+S1\&\overline{S0}\&X3+S1\&S0\&X4$

Рис.5. Структура преобразователя Simple Logic Cell

В зависимости от используемых комбинационных схем, структуры подобных функциональных преобразователей существенно отличаются друг от друга.

1.3. Организация внутренней структуры ПЛИС

В соответствии с этим критерием выделяют ПЛИС, имеющие плоскую (одноуровневую) структуру (англоязычное название - Field Programmable Gate Array - FPGA) и многоуровневую (иерархическую) структуру (Complex Programmable Logic Devices - CPLD).

Архитектура FPGA в общем виде показана на рис.6. FPGA имеет архитектуру типа ‘море вентилей’, с матрицей логических ячеек, окруженных периферийными буферами ввода/вывода.

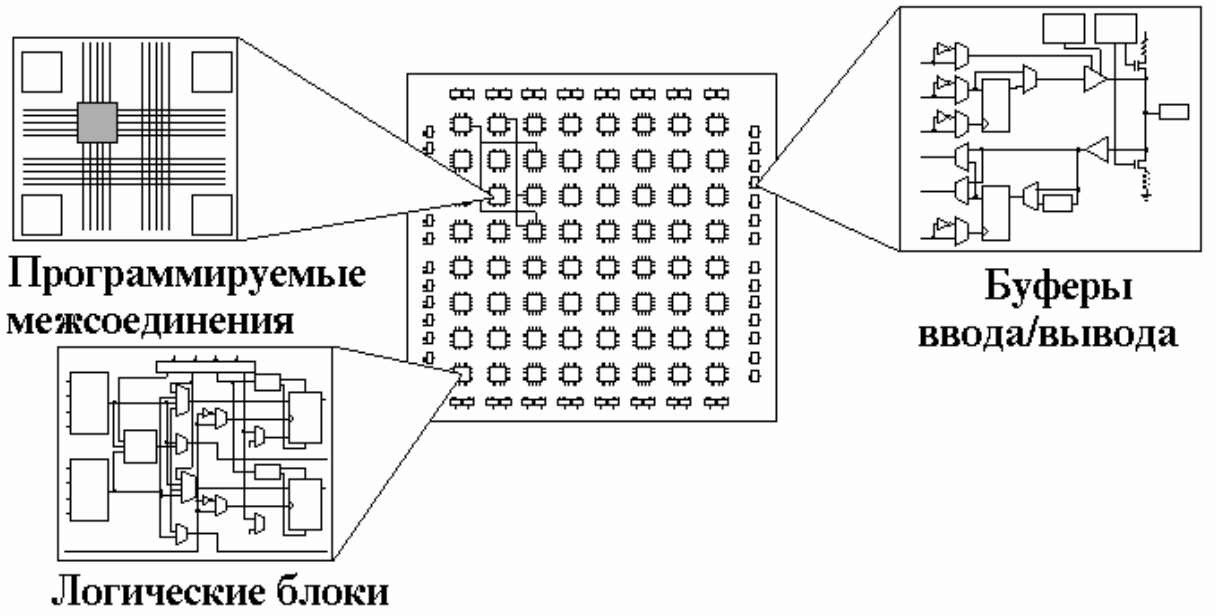


Рис.6. Архитектура FPGA

FPGA содержат простейшие функциональные преобразователи, организованные в виде матрицы или линейки, и единую для всей СБИС матрицу соединений функциональных преобразователей, разделенную узлами коммутации (рис.8). Сегменты металлических межсоединений соединяются с помощью конфигурационных элементов.

Архитектура CPLD в общем виде показана на рис.7. CPLD имеют более гибкий процесс проектирования, чем FPGA-схемы, что обусловлено особенностями их архитектуры и возможностью полной автоматизации таких этапов разработки устройства, как размещение и трассировка.

CPLD состоят из множества ПЛМ-подобных функциональных блоков (ФБ), которые могут быть соединены через матрицу межсоединений. Связь с внешними элементами схемы осуществляется через буферы ввода/вывода (БВВ). Для таких СБИС характерно наличие как глобальной матрицы соединений (ГМС) - матрицы соединений ФБ, так и локальных матриц соединений (ЛМС) - матриц соединения функциональных преобразователей.

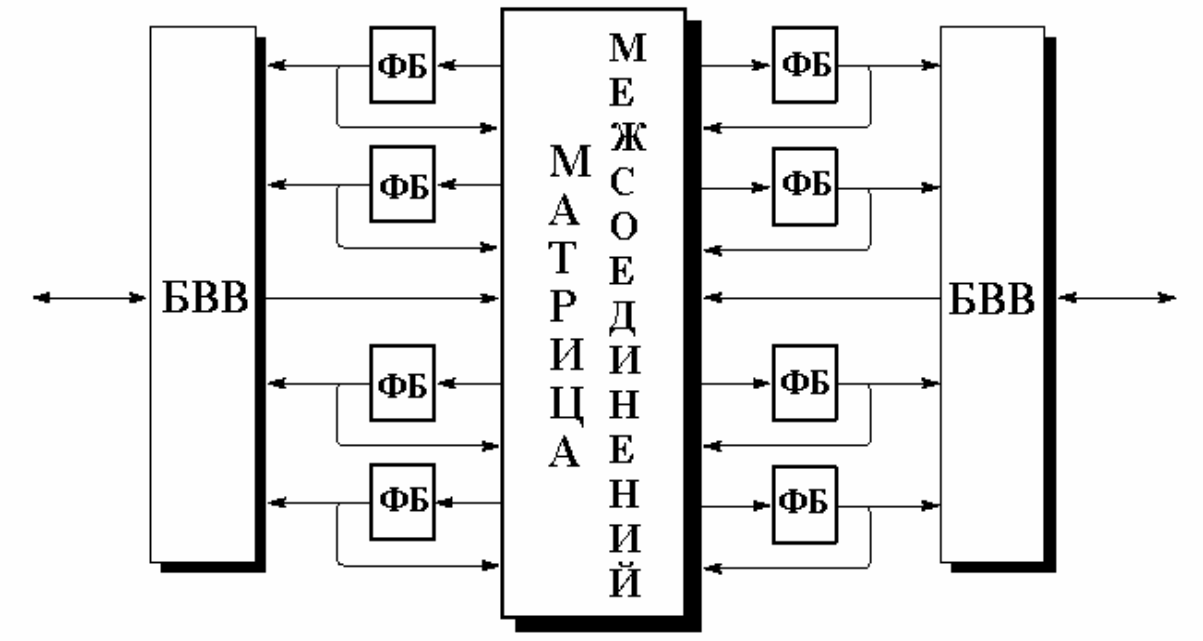


Рис.7. Архитектура CPLD

Структурная организация ПЛИС определяет особенности построения матрицы (или матриц) соединения и её основные характеристики (рис.8).

Наиболее эффективным способом выполнения соединения функциональных преобразователей, обеспечивающим минимальную и хорошо предсказуемую задержку распространения сигнала, является использование выделенного для каждого соединения непрерывного канала.

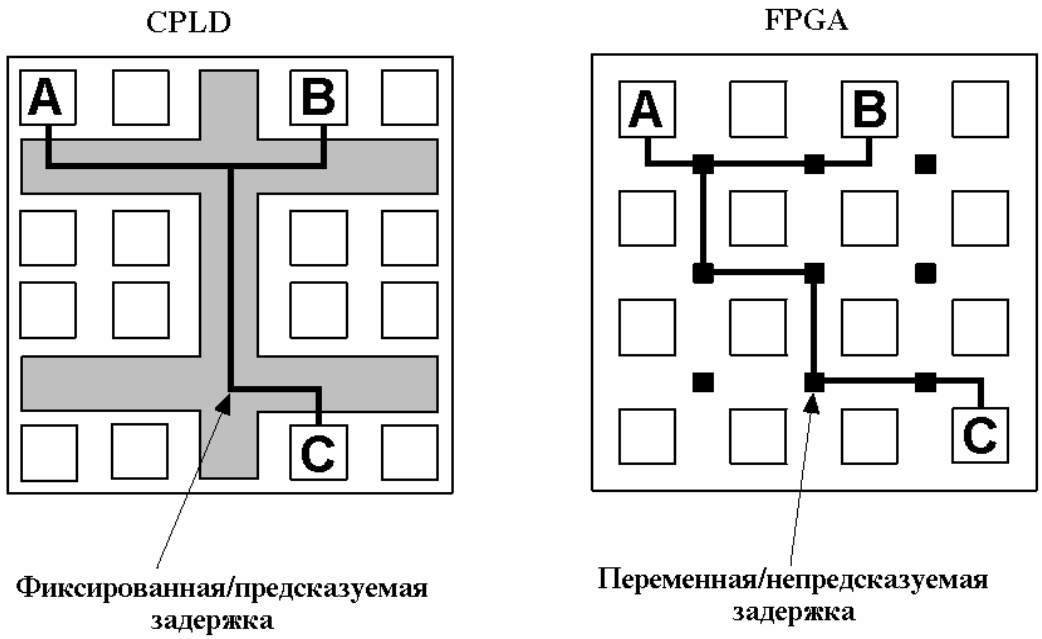


Рис.8. Способы организации внутренней структуры ПЛИС

Однако, для ПЛИС с плоской структурой (FPGA), в которых необходимо обеспечить возможность соединения между собой до нескольких тысяч простейших функциональных преобразователей, подобный подход неэффективен, так как требует слишком большого числа проходящих через всю СБИС каналов, многие из которых будут соединять только соседние функциональные преобразователи. Поэтому в FPGA используют сегментированные матрицы

соединений, состоящие из множества коротких горизонтальных и вертикальных отрезков, связанных узлами коммутации. Недостатками такого подхода являются увеличение задержки распространения сигнала, что обусловлено наличием узлов коммутации, а также непредсказуемостью задержки и её зависимость от выбранной трассы соединения.

В многоуровневых ПЛИС число функциональных преобразователей обычно невелико и расположены они компактно, поэтому локальные матрицы соединений являются непрерывными, т.е. содержат непрерывные каналы, обеспечивающие соединение функциональных преобразователей в рамках логического блока. Глобальная матрица соединений является либо полностью непрерывной, если число логических блоков невелико, либо одномерно непрерывной, т.е. непрерывной по строкам или по столбцам. В целом такую структуру организации называют непрерывной структурой соединений.

1.4. Наличие внутренней RAM-памяти

Существует два подхода к реализации в ПЛИС внутренней RAM- памяти:

- использование встроенных, крупных модулей памяти объёмом 2 Кбит;
- использование распределённых по кристаллу мелких модулей памяти объёмом порядка 32 бит.

В соответствии с первым подходом в процессе изготовления реализуется несколько (единицы и десятки) крупных модулей памяти, имеющих реконфигурируемую структуру и все необходимые элементы управления, включая синхронные регистры для хранения входных, выходных и управляющих сигналов. Так как при таком подходе модули памяти занимают отдельную, выделенную часть площади кристалла, то, независимо от используемого объёма памяти, число доступных разработчику простейших функциональных преобразователей не уменьшается.

Другой подход предполагает использование простейших функциональных преобразователей для реализации модулей памяти объёмом 16×2 бит либо 32×1 бит. При этом, при построении модулей памяти большого объёма уменьшается число доступных разработчику функциональных преобразователей, снижается их быстродействие, что обусловлено задержками распространения сигналов в сегментированной матрице соединений.

2. ТЕХНОЛОГИЯ ИЗГОТОВЛЕНИЯ КОНФИГУРАЦИОННЫХ ЭЛЕМЕНТОВ ПЛИС

Как было показано, ПЛИС состоит из некоторого количества логических модулей одного или нескольких типов. В процессе программирования модули конфигурируются для выполнения определенной функции, а также соединяются между собой для реализации задуманной схемы. Это осуществляется с помощью сегментов межсоединений и программируемых (конфигурационных) элементов. Основными технологиями изготовления конфигурационных элементов являются: EPROM - программируемые элементы допускают ультрафиолетовое стирание; EEPROM - программируемые элементы допускают электрическое стирание; FLASH - программируемые элементы допускают ускоренную электрическую запись (перезапись); SRAM - программируемые элементы реализованы на статических запоминающих ячейках; Antifuse - программируемые элементы реализованы на однократно программируемых, исходно разомкнутых перемычках.

Технология SRAM обеспечивает возможность выполнения неограниченного числа циклов конфигурирования ПЛИС. Указанное свойство полезно на этапе отладки, а также позволяет путём загрузки новой конфигурации изменять алгоритм работы ПЛИС без выключения питания. Однако, поскольку после выключения питания ПЛИС на SRAM ячейках теряет информацию о конфигурации, то после каждого выключения питания необходимо выполнить цикл конфигурирования из внешнего источника хранения конфигурирующих данных (ПЗУ).

Технологии FLASH и EEPROM допускают выполнение до 10 000 и 100 циклов соответственно.

ПЛИС, выполненные по технологии EPROM, в настоящее время являются однократно программируемыми, так как для обеспечения их репрограммируемости вместо дешевого пластмассового корпуса требуется использовать дорогой керамический корпус.

2.1. Конфигурационный элемент EPROM

EPROM-транзистор - это модифицированный NMOП-транзистор, в котором пороговое напряжение легко переключается между низким уровнем (ниже нуля (Vss)) и высоким уровнем (больше единицы (Vcc)). Различные пороговые напряжения переводят EPROM-ячейку в состояния включено/выключено.

EPROM-транзистор имеет плавающий поликремниевый затвор, расположенный между затвором доступа (access gate) и подложкой, как показано на рис.9.

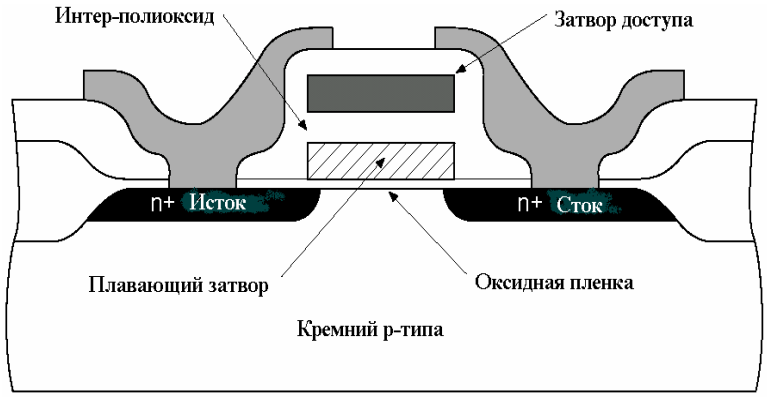


Рис.9. EPROM-транзистор

Плавающий затвор электрически изолирован от подложки тонким слоем (примерно 200 А) оксида, а от затвора доступа (ЗД) более толстым слоем диэлектрического интер-полиоксида, который обычно состоит из оксидов и/или нитридов.

EPROM-транзистор программируется высоким уровнем напряжения с горячей инжекцией электронов (рис.10). Когда высокий уровень напряжения (Vpp) прикладывается к ЗД EPROM-ячейки, а незначительно меньшее напряжение (Vd) прикладывается к его стоку, электроны движутся от истока к стоку. С повышением кинетической энергии электронов их путь изменяется электрическим полем, расположенным между ЗД и подложкой. Это электрическое поле создается разностью потенциалов между ЗД (Vpp) и стоком (Vd).

Электроны, достигшие кинетической энергии 3.2 eV, устремляются по направлению к плавающему затвору, проходя через тонкую оксидную пленку, отделяющую затвор от подложки, и попадают на плавающий затвор.

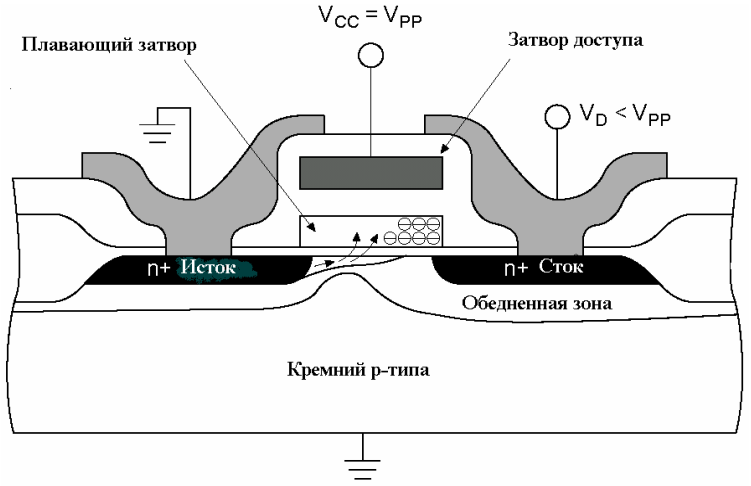


Рис.10. Программирование EPROM-ячейки

Эти электроны создают отрицательный заряд на плавающем затворе, который противодействует электрическому полю, созданному положительным напряжением на ЗД. Результатом является существенное увеличение порогового напряжения, требуемого для перевода EPROM-ячейки из непроводящего в проводящее состояние.

На рис.11 показаны вольт-амперные характеристики (ВАХ) для запрограммированной (высокое пороговое напряжение) и стертой (низкое пороговое напряжение) EPROM-ячейки.

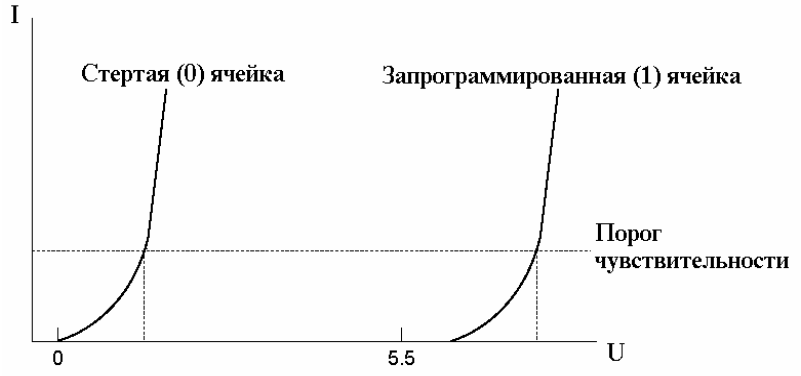


Рис.11. ВАХ EPROM-ячейки

Запрограммированная EPROM-ячейка ведет себя как транзистор в закрытом состоянии. Ток исток-сток в этом случае не течет из-за напряжений затвора доступа, изменяющихся от 0 до Vcc. И наоборот, через стертую ячейку протекает ток исток-сток, когда напряжение на затворе доступа равно примерно 1V, подобно открытому транзистору.

Стирание запрограммированных EPROM-ячеек производится ультрафиолетовым излучением с длиной волны 2.540 A. Избыточные электроны на плавающем затворе поглощают энергию ультрафиолетового излучения, в результате чего их энергетический уровень становится достаточным для преодоления барьера в 3.2 eV. В результате электроны мигрируют в подложку, где и нейтрализуются.

2.2. Конфигурационный EEPROM

EEPROM-транзистор (также, как и EPROM) - это МОП-транзистор, который включается/выключается в зависимости от величины порогового напряжения. Однако в отличие от EPROM-схем EEPROM-схемы могут программироваться электрически. EEPROM-ячейка построена на поликремниевой структуре с плавающим затвором (рис.12).

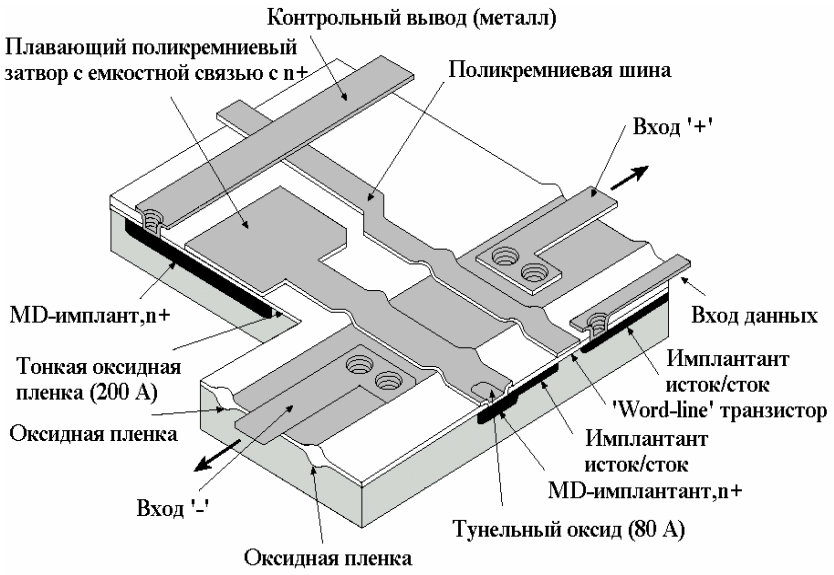


Рис.12. Конструкция EEPROM-ячейки

Пороговое напряжение меняется, когда туннельный механизм создает избыток электронов на плавающем затворе. Туннельный механизм начинает работать, когда плавающий затвор заряжается до высокого напряжения (12...13 V) через емкостное соединение в диффузионной области n+. Как только электроны попали на плавающий затвор, они создают отрицательное электрическое поле, тем самым увеличивая пороговое напряжение транзистора, и препятствуют переключению транзистора при напряжениях ниже определенного уровня. Этот процесс позволяет плавающему затвору выступать в качестве переключателя (включено/выключено) транзистора.

EEPROM-ячейка стирается с помощью того же туннельного механизма, что и EPROM-ячейка. Когда на плавающем затворе электронов нет (затвор имеет положительный заряд),

включение/выключение EEPROM-транзистора происходит в зависимости от напряжения на контрольном входе.

На рис.13 показана двухэлектродная структура, в которой один электрод сформирован на поликремнии, а другой - на сильно легированном поликремнии n-типа.

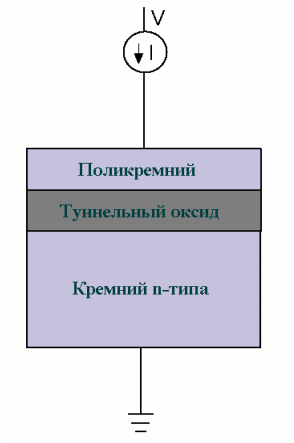


Рис.13. Структура EEPROM-ячейки

Электроды разделены туннельным оксидом с толщиной примерно 80 А. Когда обычное рабочее напряжение (5V или менее) приложено к туннельному оксиду, он действует как диэлектрик и не проводит электрический ток, причем туннельный ток имеет предельно малое значение (менее 10^{-20} А). Однако, когда прикладывается высокое напряжение (12...14 V), электроны проходят через оксид. Высокие напряжения используются для программирования/стирания ячейки (т.е. заряда/разряда плавающего затвора), при этом ток через туннельный оксид достигает 1мкА. В зависимости от полярности напряжения этого тока достаточно, чтобы зарядить/разрядить ячейку за несколько миллисекунд.

ВАХ запрограммированной и стертой EEPROM-ячейки практически полностью совпадают с ВАХ EPROM-ячейки. Однако пороговое напряжение разряженной EEPROM-ячейки - отрицательное (менее 0V), потому что на плавающем затворе наблюдается недостаток электронов.

2.3. Конфигурационный элемент FLASH

FLASH-транзистор сочетает в себе конструкцию, технологию и рабочие характеристики EPROM- и EEPROM-транзисторов. Как показано на рис.14, FLASH-транзистор имеет два слоя поликремния в структуре затвора, которая похожа на структуру EPROM-транзистора. Однако транзисторы имеют различные толщины слоев затвор-оксид и исток/стоковых областей: толщина затворного оксида у FLASH-ячейки менее 100 А, тогда как у EPROM-ячейки - 200 А.

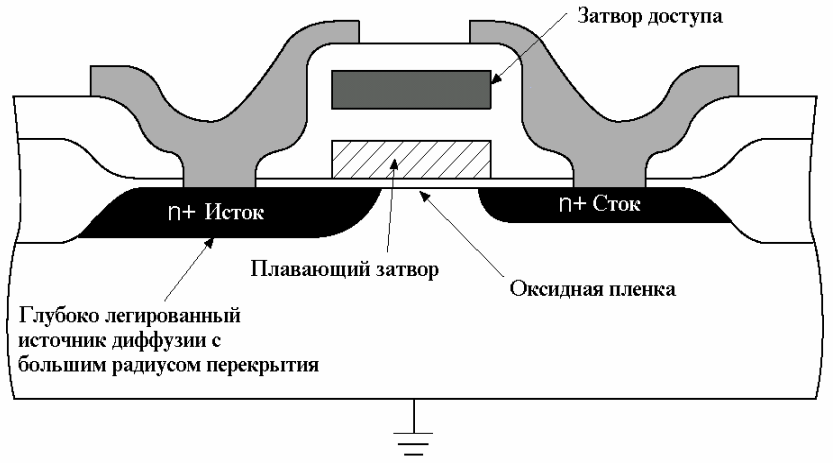


Рис.14. FLASH-транзистор

Диффузионные области истока и стока FLASH-ячейки асимметричны; исток имеет больший коэффициент диффузии, чем сток. Перекрытие исток-затвор больше перекрытия сток-затвор: образуется градиентная диффузия с большим напряжением пробоя.

Затвор доступа покрыт вольфрамовой пленкой, что эффективно снижает сопротивление второго слоя поликремния и улучшает характеристики прибора. Вольфрамовая пленка не влияет на работу FLASH-ячейки.

FLASH-транзистор, как и EPROM, программируется с помощью горячей инжекции электронов (рис.15).

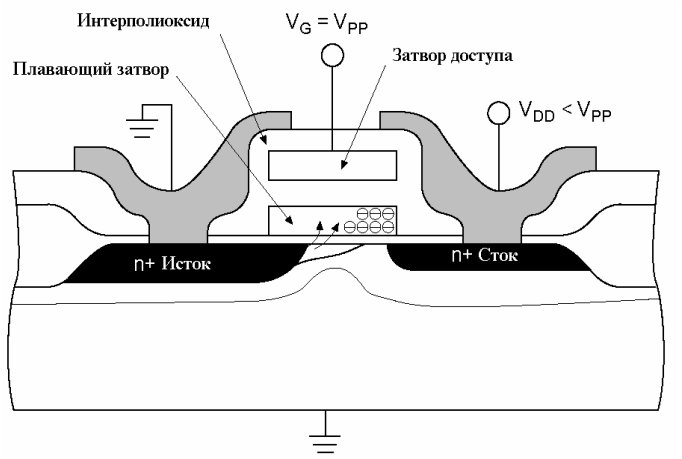


Рис.15. Программирование FLASH-транзистора

Когда высокое напряжение программирования Vpp (примерно 12V) приложено к затвору доступа, происходит емкостное соединение его с плавающим затвором: ячейка включается. Низкое напряжение Vdd (5...8 V), приложенное к стоку, вызывает большой ток исток-сток. Электрическое поле, созданное соединенными затвором доступа и плавающим затвором, отклоняет поток электронов, как показано на рис.10. Отклоненные электроны с энергией более 3.2 eV проникают через оксидную пленку и захватываются поликремниевым плавающим затвором. После снятия напряжения программирования электроны, пойманные плавающим затвором, поднимают пороговое напряжение выше 5.5 V, выключая ячейку.

Чтобы стереть FLASH-транзистор, необходимо удалить избыток электронов с плавающего затвора (туннельный эффект). Для этого заземляется затвор доступа, тем самым заземляя и плавающий затвор, а высокое напряжение программирования Vpp прикладывается к истоку, как показано на рис.16.

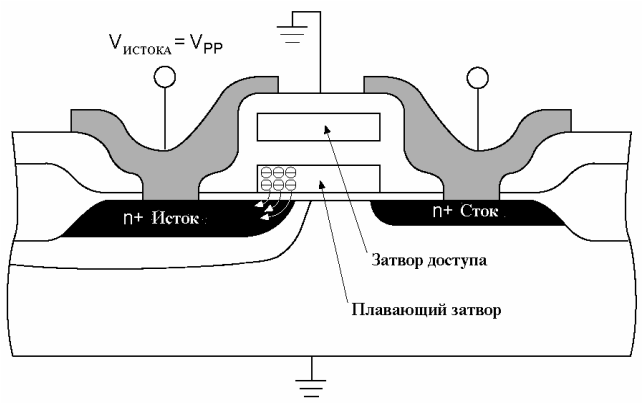


Рис.16. Стирание FLASH-транзистора

В результате создается сильное электрическое поле (примерно $12 \cdot 10^6$ V/cm) между плавающим затвором и истоковым переходом, образующее туннельный ток, который разряжает ячейку.

2.4. Конфигурационный элемент SRAM

На рис.17 показана стандартная КМОП пятитранзисторная ячейка, которая и составляет конфигурационный элемент SRAM. Технологический процесс изготовления SRAM-приборов является разновидностью процесса изготовления EEPROM-приборов.

2.5. Конфигурационный элемент ANTIFUSE

Все рассмотренные выше соединения (EPROM, EEPROM, FLASH и SRAM) являются перепрограммируемыми. Существует довольно большой класс микросхем, основанных на неперепрограммируемых соединениях типа Antifuse (Все микросхемы ф. Actel, серия XC8100 ф. Xilinx).

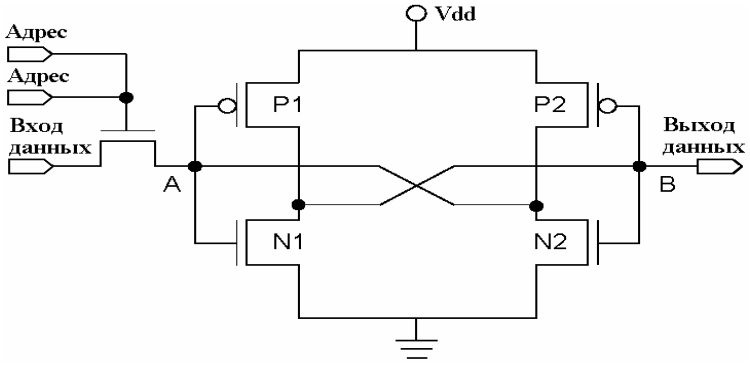


Рис.17. Конфигурационный элемент SRAM

Для получения соединения используется структура, показанная на рис. 18. Изначально электрического соединения между сегментами нет. Когда между слоями металла прикладывается достаточно высокое напряжение, слой диэлектрика плавится, обеспечивая протекание тока. Antifuse-соединение по размеру

меньше, чем ширина сегмента металлизации, имеет малое сопротивление (100...600 Ом) и емкость (5...13ФФ). Это позволяет в несколько раз увеличить по сравнению с традиционными технологиями количество сегментов межсоединений, что обеспечивает лучшие трассировочные возможности.

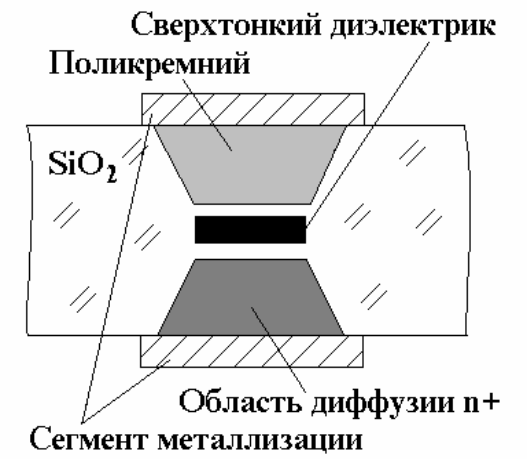


Рис.18. Конструкция элемента ANTIFUSE

Отсутствие возможности перепрограммирования не позволяет использовать эти микросхемы в проектах, где без перепрограммирования нельзя обойтись, а также там, где необходимо протестировать оборудование или внести коррективы.

3. ОБЛАСТИ ПРИМЕНЕНИЯ ПЛИС

В первые годы развития технологии ПЛИС сложилось ошибочное мнение, что CPLD-архитектура оптимальна только для комбинационных схем (дешифраторы, мультиплексоры, сумматоры, компараторы), а FPGA архитектура - для последовательностных схем (триггеры, счетчики, сдвигающие регистры и т.д.). На рис.19 показаны наиболее подходящие варианты применений для CPLD- и FPGA-архитектур.

CPLD используют несколько разновидностей КМОП-технологии и вариантов архитектуры. Например, EPROM-, EEPROM- и FLASH-схемы, таких семейств, как Classic, MAX5000, MAX7000, MAX9000, FLASHlogic ф.Altera, имеют архитектуру, оптимизированную под проекты с комбинационной насыщенностью, а SRAM-CPLD-семейств FLEX8000 (ф.Altera) и XC2000, XC3000, XC4000 (ф.Xilinx) используют блочную архитектуру, оптимизированную под проекты с высокой регистровой насыщенностью. SRAM-приборы обладают способностью внутрисистемного репрограммирования.

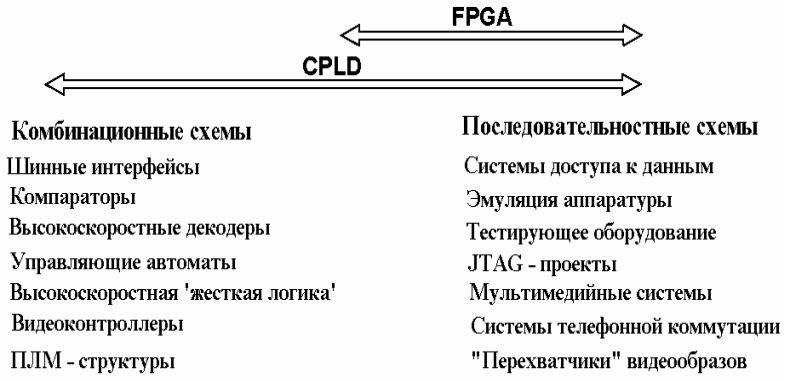


Рис.19. Сферы применения ПЛИС

В отличие от CPLD-схем большинство FPGA-схем использует только SRAM или однократнопрограммируемые Antifuse элементы памяти. Блочная архитектура FPGA обеспечивает широкий круг применений, однако менее эффективна, чем CPLD при реализации проектов с высокой комбинационной насыщенностью.

3.1. Достоинства и недостатки ПЛИС

К достоинствам ПЛИС следует отнести:

- сокращение времени изготовления;
- снижение стоимости разработки;
- для ПЛИС, не использующих antifuse-элементы возможность перепрограммирования, реконфигурирования и отладки непосредственно на плате (JTAG-интерфейс);
- высокая надежность;
- сохранение интеллектуальных свойств проекта, закрытость проекта от копирования;
- эффективность использования при мелкосерийном и единичном производстве;
- возможность проведения всего цикла проектирования и конфигурирования (программирования) на одном рабочем месте.

К недостаткам можно отнести:

- сравнительно (с заказными БИС и БМК) невысокую рабочую скорость;
- сравнительно невысокую плотность упаковки (значительную площадь кристалла ПЛИС занимают межсоединения);
- сравнительно высокую стоимость.

3.2. Перспективные направления развития ПЛИС

За последние годы резко возросла логическая емкость (количество вентилей на кристалле) ПЛИС. Так, выпускаемые по SRAM (Sequel Random Access Memory) технологии FPGA фирм Altera и Xilinx перешагнули рубеж в 1 млн эквивалентных вентилей 2И-НЕ на кристалле.

ПЛИС становятся основой для «систем на кристалле» (system-on-chip, SOC). В основе идеи SOC лежит интеграция всей электронной системы в одном кристалле. Компоненты этих систем разрабатываются отдельно и хранятся в виде файлов параметризируемых модулей. Благодаря стандартизации в одно целое можно объединять модули разных разработчиков.

По существу, на площади одной кремниевой подложки требуется разместить не только миллионы узлов базисной логики, но и процессорное ядро с набором разнообразных периферийных модулей. То есть все то, что находится на типовой системной плате. В самом общем случае она содержит процессор (DSP – digital signal processor, микроконтроллер или микропроцессор), специализированную логику (ASSP – Application Specific Standart Products) в виде вентильных матриц (Gate Arrays) или заказных микросхем (ASIC) и программируемую логику – ПЛИС. Кроме этого имеются блоки энергонезависимой и статической памяти, периферийные устройства, аналоговые узлы и модули управления питанием.

Примером новых семейств ПЛИС, пригодных для реализации «систем на кристалле», является семейство APEX20K фирмы Altera. Архитектура APEX20K сочетает в себе как достоинства FPGA с их таблицами перекодировок (LUT), входящими в состав логического элемента ПЛИС, так и логику вычисления СДНФ (сокращенная нормальная дизъюнктивная форма), характерную для CPLD. Таким образом, новые ПЛИС пригодны как для решения задач цифровой обработки сигналов, так и для реализации сложных логических автоматов. Среди других семейств ПЛИС большой емкости следует отметить семейство Virtex фирмы Xilinx, семейство ProASIC фирмы Actel, отличительной особенностью которого является энергонезависимость интегрированного на кристалле запоминающего устройства благодаря применению FLASH-технологии.

Рассмотрим более детально перспективы реализации идеи SOC на примере нового семейства FPSLIC фирмы Atmel. Концепцию FPSLIC можно рассматривать как первый шаг к слиянию двух путей развития сложных универсальных микросхем (микропроцессоров и программируемой логики) в единое целое. В основу нового кристалла, выполненного по технологии 0.35 мкм, положены массив FPGA и AVR-микроконтроллер Atmega161. Здесь впервые стандартное ядро AVR выполняет команды из SRAM, что значительно повышает скорость его работы (тактовая частота до 40 МГц). Для дополнительной эффективности при выполнении DSP-приложений к ядру AVR добавлен аппаратный 8×8 умножитель, дающий 16-разрядный результат. На кристалле FPSLIC также размещен набор фиксированных периферийных улов: два UART, три таймера-счетчика (два 8-разрядных и один 16-разрядный) и два порта ввода/вывода. Добавлен аппаратный интерфейс I2C, позволяющий AVR обмениваться данными с внешней конфигурационной EEPROM, которая используется для программирования FPSLIC. Блок фиксированной логики, размещенной между AVR и FPGA, позволяет использовать массив FPGA для реализации дополнительных программируемых периферийных узлов в реальном проекте, причем эти

новые периферийные устройства будут доступны в общем адресном пространстве памяти AVR. Архитектура массива статической памяти SRAM внутри кристалла FPSLIC реализована так, чтобы обеспечить разработчику максимальную гибкость в распределении адресного пространства. Размер памяти программ составляет 10К×16, памяти данных - 4К×8. Помимо этих фиксированных массивов на кристалле имеется еще один блок памяти 6К×16, который может использоваться или как дополнительная память программ, или как дополнительная память данных, в зависимости от решаемой задачи.

В FPSLIC реализована возможность создания динамически реконфигурируемой системы, поскольку FPGA может быть перепрограммирована под управлением AVR непосредственно в процессе работы. Это обстоятельство является очень важным, учитывая требования современного рынка носимых реконфигурируемых систем. Реконфигурируемые системы позволяют заметно снижать потребление энергии, что является сильным аргументом для использования технологии FPSLIC при создании портативных и носимых «интеллектуальных устройств».

Семейство FPSLIC, содержащее AVR и FPGA, имеет обозначение AT94Kxx. Новые кристаллы полностью совместимы по расположению и назначению внешних выводов с микросхемами FPGA семейств Atmel At40K, Xilinx 4000, 5200 и SPARTAN и являются также PCI-совместимыми.

Идея, реализованная корпорацией Atmel, пока является наиболее удачной. За разработку FPSLIC руководители направлений SOC и FPGA – Мартин Мейсон и Джоэл Розенберг – были удостоены приза США «За лучший проект 1999 года».

3.3. Обзор семейств ПЛИС фирмы Altera

Фирма Altera является крупнейшим мировым производителем ПЛИС. В настоящее время наибольшее распространение получили шесть семейств ПЛИС (табл.2), поддерживаемых САПР MAX PLUS II.

Семейство Classic объединяет три серии ПЛИС. ПЛИС этого семейства позволяют заменить устройство, содержащее от 10 до 20 микросхем средней степени интеграции и обеспечивают:
- задержку распространения сигнала от любого входа до выхода БИС не более 10 нс;
- устойчивую работу на частотах до 100 МГц;
- возможность работы в режиме пониженного энергопотребления (Turbo-off), позволяющего сократить потребление энергии до уровня единиц мА при частотах до 500 кГц и до уровня единиц мкА при нулевой тактовой частоте;
- возможность задания режима секретности разработки.

Таблица 2

ХАРАКТЕРИСТИКИ СЕМЕЙСТВ ПЛИС ФИРМЫ ALTERA

MAX7000 (E)S	MAX9000	FLEX8000	FLEX10K
600-5000	6000-12000	2500-16000	10000-100000
-	-	-	Есть
36-164	59-216	68-208	59-406
EEPROM	EEPROM	SRAM	SRAM

Семейство / Хар-ки	Classic	MAX5000
Логическая емкость	300-900	600-3750
Наличие внутренней памяти	-	-
Число доступных выводов	22-64	24-84
Технология	EPROM	EPROM

Одноуровневая структура ПЛИС семейства Classic включает единую для всей СБИС программируемую матрицу "И" и набор макроячеек (МЯ) - простейших функциональных преобразователей, имеющих классическую GAL-архитектуру. Выводы МЯ жестко связаны с выводами корпуса.

Семейство MAX (Maltiple Array matriX) 5000 объединяет пять серий СБИС. СБИС этого семейства позволяют заменить устройство, содержащее до нескольких десятков микросхем средней степени интеграции и обеспечивают:

- задержку распространения сигнала от любого входа до выхода не более 15 нс;
- устойчивую работу на частотах до 76 МГц;
- возможность задания режима секретности разработки;
- возможность использования трех режимов работы выходных буферов: вход, выход, двунаправленный.

Двухуровневая структура ПЛИС семейства MAX5000 включает: логические блоки (ЛБ), содержащие 16 макроячеек с расширенной GAL- архитектурой и локальную программируемую матрицу "И"; программируемую матрицу соединений с непрерывной структурой. Наличие программируемой матрицы соединений обеспечивает большую, по сравнению с семейством Classic, гибкость в размещении внутренних ресурсов и выводов СБИС.

Семейство MAX 7000 объединяет семь серий СБИС. СБИС этого семейства позволяют заменить устройство, содержащее до сотни корпусов микросхем средней степени интеграции, и обеспечивают:

- задержку распространения сигнала от любого входа до выхода не более 5 нс;
- устойчивую работу на частотах до 178 МГц;
- возможность регулирования скорости переключения выходных буферов;
- возможность использования четырех режимов работы выходных буферов: вход, выход, двунаправленный, открытый коллектор;
- возможность задания режима пониженного энергопотребления (Turbo-off) как для всей СБИС, так и для цепей распространения отдельных сигналов;
- возможность программирования/репрограммирования после распайки на плате (JTAG-интерфейс);
- возможность задания режима секретности разработки;
- работу с пониженным напряжением питания (3.3 В).

Кроме того, СБИС ряда серий семейства MAX7000 соответствуют требованиям стандарта шины PCI.

Двухуровневая структура СБИС семейства MAX7000 включает: логические блоки (ЛБ), содержащие 16 макроячеек с архитектурой ПЛУ и локальную программируемую матрицу "И"; улучшенную программируемую матрицу соединений (ПМС) с непрерывной структурой; программируемые модули ввода/вывода.

Расширенные коммутационные возможности ПМС и наличие программируемых модулей ввода/вывода, отделяющих выход макроячейки от вывода СБИС, обеспечивают большие, по сравнению с рассмотренными ранее семействами, возможности разводки кристалла и управления выводами.

Семейство FLEX (Flexible Logic Element matriX) 8000A объединяет шесть серий СБИС. СБИС этого семейства позволяют заменить устройство, занимающее десятки плат, выполненных на микросхемах средней степени интеграции и обеспечивают возможность:

- устойчивой работы на частотах до 294 МГц;
- эмуляции внутренней шины с тремя состояниями;
- работы с пониженным напряжением питания (3.3 В);
- работы в системах со смешанным напряжением питания (3.3 В, 5.0 В);
- реализации неограниченного числа циклов конфигурирования, в том числе без выключения питания СБИС;

- регулирования скорости переключения выходных буферов;
- использования трех режимов работы выходных буферов: вход, выход, двунаправленный.

Кроме того, все СБИС этого семейства совместимы со стандартом шины PCI.

Двухуровневая структура СБИС семейства FLEX8000 включает: логические блоки (ЛБ), содержащие 8 логических элементов (ЛЭ) с табличной архитектурой и имеющие локальную программируемую матрицу соединений с непрерывной структурой связей; глобальную программируемую матрицу соединений с одномерно-непрерывной структурой (непрерывной по строкам и столбцам); программируемые элементы ввода/вывода с синхронным триггером. Отличительной особенностью структуры СБИС данного семейства является то, что элементы ввода/вывода соединяются не с выводами ЛЭ, а с вертикальными и горизонтальными каналами глобальной программируемой матрицы соединений. Такое архитектурное решение обеспечивает исключительную гибкость в распределении внутренних логических ресурсов и размещении выводов СБИС. Фактически разработчик, дорабатывая уже распаянную на плате СБИС, может полностью изменить логику ее работы, сохраняя размещение выводов.

Семейство MAX 9000 объединяет четыре серии микросхем. СБИС этого семейства позволяют заменить устройство, занимающее десятки плат, выполненных на микросхемах средней степени интеграции, и обеспечивают возможность:
- устойчивой работы на частотах до 125 МГц;
- независимого использования логической части и триггера макроячейки;
- задания режима пониженного энергопотребления (power-saving mode) как для всей СБИС в целом, так и для распространения отдельных сигналов;
- программирования и репрограммирования после распайки на плате;
- работы в системах со смешанным напряжением питания (3.3 В, 5.0 В);
- регулирования скорости переключения выходных буферов;
- использования трех режимов работы выходных буферов: вход, выход, двунаправленный;
- все серии ПЛИС семейства MAX9000 совместимы со стандартом PCI.

Двухуровневая логика СБИС семейства MAX9000 включает: логические блоки, содержащие 16 макроячеек с программируемой архитектурой и имеющие локальную программируемую матрицу соединений с одномернонепрерывной структурой (непрерывной по строкам и по столбцам); программируемые элементы ввода/вывода с синхронным триггером.

Семейство FLEX (Flexible Logic Element matriX) 10K объединяет семь серий СБИС. СБИС этого семейства позволяют заменить устройство, занимающее сотни плат, выполненных на микросхемах средней степени интеграции, и обеспечивают возможность:
- устойчивой работы на частотах до 425 МГц;
- реализации на кристалле статической памяти и ПЗУ объемом до 24 Кбит;
- независимого использования логической части и триггера каждого логического элемента;
- эмуляции внутренней шины с тремя состояниями;
- умножения внутренней тактовой частоты;
- работы в системах со смешанным напряжением питания (3.3 В, 5.0 В);
- реализации неограниченного числа циклов конфигурирования;
- регулирования скорости переключения выходных буферов;
- использования четырех режимов работы выходных буферов: вход, выход, двунаправленный, открытый коллектор;
- все СБИС семейства FLEX10K совместимы со стандартом шины PCI.

Двухуровневая логика СБИС этого семейства включает: логические блоки, содержащие 8 логических элементов с табличной архитектурой и имеющие локальную программируемую матрицу соединений с непрерывной структурой связей; встроенные реконфигурируемые модули памяти, позволяющие реализовывать как статическую память и ПЗУ, так и сложные логические функции; глобальную программируемую матрицу соединений с одномерно непрерывной структурой (непрерывной по строкам и по столбцам); программируемые элементы ввода/вывода с синхронным триггером.

4. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Лабораторный практикум предназначен для закрепления знаний, полученных студентами на лекциях, а также для приобретения навыков работы с САПР ПЛИС и микросхемами программируемой логики. Практикум состоит из пяти лабораторных работ:

1) графический редактор системы MAX PLUS II;
2) работа с компилятором системы MAX PLUS II;
3) **логический синтез проекта на ПЛИС;**
4) **моделирование и временной анализ проекта на ПЛИС;**
5) программирование ПЛИС.

В первой работе изучается общий интерфейс САПР ПЛИС, основные команды графического редактора.

Вторая и третья работы посвящены изучению компилятора системы MAX PLUS II. Во второй работе основное внимание уделяется изучению общего интерфейса, главного меню компилятора. В третьей работе подробно изучается этап логического синтеза и опция ‘Global Project Logic Syntesis’.

В четвертой работе рассматриваются вопросы моделирования и временного анализа в современных САПР ПЛИС. Особое внимание уделяется сравнительному анализу методов моделирования.

Пятая работа имеет прикладной характер: в процессе выполнения работы производится конфигурация (“прошивка”) ПЛИС и сравнение практических результатов с результатами моделирования.

4.1. Лабораторная работа №1. Графический редактор системы MAX PLUS II

Цель работы: ознакомление с маршрутом проектирования ПЛИС, способами входного описания проекта, общим пользовательским интерфейсом системы MAX PLUS II (Multiple Array Matrix Programmable Logic User System), получение навыков работы с графическим редактором системы MAX PLUS II.

Введение

Традиционный процесс проектирования ПЛИС состоит из следующих этапов:
- ввод проекта;
- компиляция;
- верификация;
- программирование.

Ввод проекта может осуществляться несколькими способами с использованием:
- таблицы истинности;
- булевых уравнений;
- временных диаграмм;
- электрических схем;
- языков описания высокого уровня;
- конечных автоматов;
- назначением ножек, внутренних ячеек, блоков микросхемы.

Таблицы истинности и булевы уравнения используются для описания небольших проектов, так как при большом количестве входных и внутренних переменных табличное описание становится громоздким и неудобным, размерность задачи резко возрастает.

Описание с помощью временных диаграмм не пользуется большой популярностью, потому что не позволяет оптимально задавать ограничения и требования к проектируемому устройству: САПР самостоятельно решает, каким образом реализовать заданные входные воздействия.

Схемное описание проекта позволяет с помощью набора библиотечных элементов (обычно это микросхемы серий 1533 и 555) и макрофункций (более сложные элементы, например, RAM, ROM, ALU, порты ввода/вывода и т.д.) задавать алгоритм функционирования проектируемого устройства. Для схемного описания характерна большая вложенность (иерархичность) структуры: каждый элемент на схеме может представлять отдельную электрическую схему или текстовое описание (например на языке высокого уровня). В настоящее время широкое распространение получает такой метод проектирования, при котором блоки устройства задаются в виде элементов, а внутреннее описание элементов - на языке высокого уровня (VHDL, Verilog, ABEL и т.д.).

Языки высокого уровня в настоящее время пользуются наибольшей популярностью. Многие из них (VHDL, Verilog) являются международными стандартами проектирования аппаратуры. Проект, описанный на языке высокого уровня, может без каких-либо изменений и корректировок передаваться между различными пакетами САПР и разработчиками. Кроме того, во многих странах разработчики радиоэлектронной аппаратуры и микросхем обязаны поставлять в составе технической документации модели на языках VHDL и Verilog. В связи с этим многие фирмы-разработчики вообще отказались от обязательного ранее схемного представления своих проектов.

Описание с помощью конечных автоматов позволяет наглядно и компактно задавать и отлаживать сложные проекты. Практически любая современная САПР ПЛИС или система моделирования имеют средства для преобразования автоматного описания в текстовое на языке высокого уровня. Отладка такого проекта заключается в редактировании структуры конечного автомата, трансляции её в язык высокого уровня и моделировании.

Задание алгоритма функционирования устройства с помощью назначения логических блоков ПЛИС используется редко и только опытными разработчиками. Для этого пользуются редактором разводки, в котором в графическом виде условно представлена топология ПЛИС. Разработчик вручную вводит логические связи между логическими блоками для реализации заданной функции. В данном случае может быть получено оптимальное с точки зрения быстродействия и плотности упаковки

устройство. Существенными недостатками такого подхода являются высокая трудоёмкость и невозможность составления документации на разработанное устройство.

Подготовка к выполнению лабораторной работы

Система MAX PLUS II ф.Altera обеспечивает достаточно широкий спектр возможностей входного описания проекта. В её составе имеются следующие редакторы: Graphic Editor (графический редактор), Symbol Editor (редактор элементов схем), Text Editor (текстовый редактор, поддерживаются языки VHDL, Verilog, AHDL(Altera Hardware Description Language)), Waveform Editor (редактор временных диаграмм), Floorplan Editor (редактор разводки).

В данной лабораторной работе рассматривается графический редактор системы MAX PLUS II (Graphic Editor).

Пользовательский интерфейс системы MAX PLUS II при работе с графическим редактором показан на рис. 20.

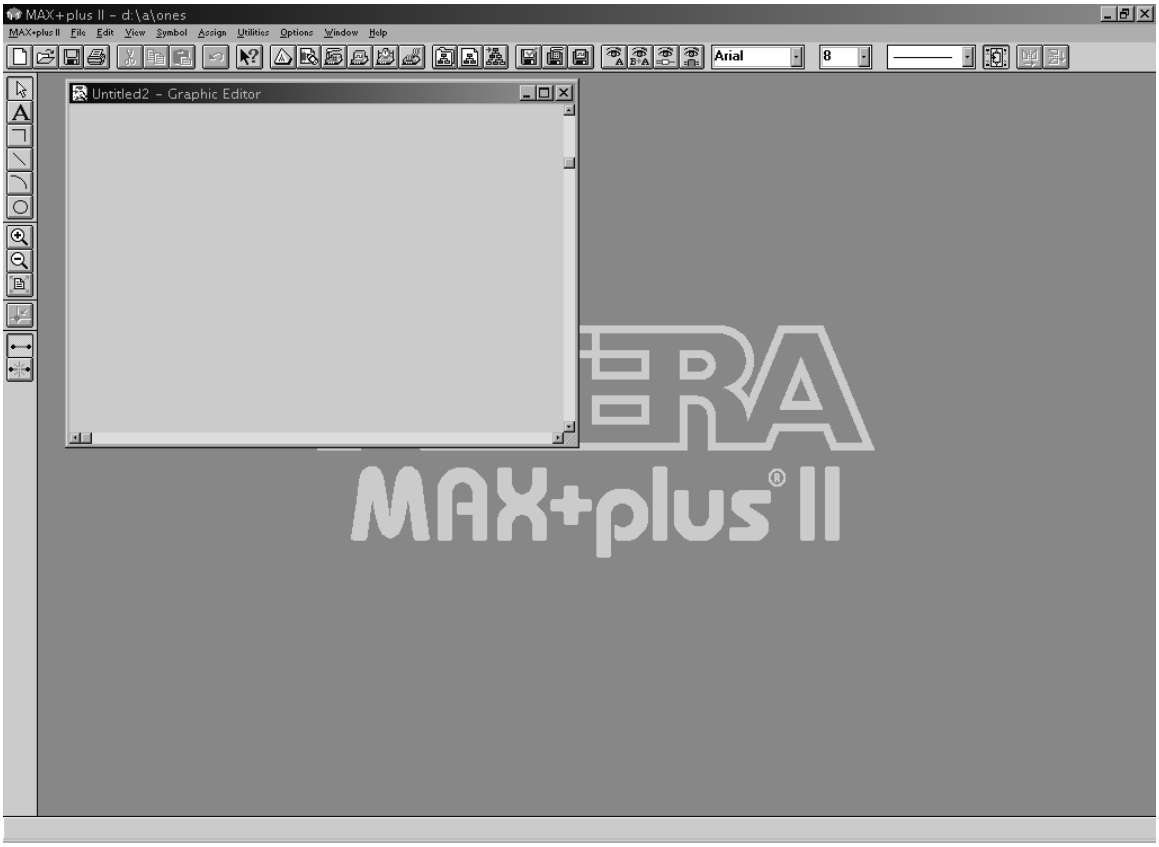


Рис.20. Пользовательский интерфейс системы MAX PLUS II

Опция ‘MAX+plus II’ основного меню позволяет открыть любое приложение системы (рис.21).

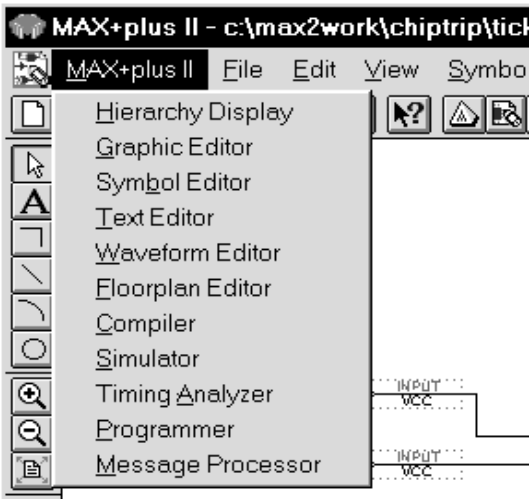


Рис.21. Опция ‘MAX+plus II’

Система MAX PLUS II обладает мощной встроенной системой помощи. В нижней части окна автоматически появляется подсказка о каждом объекте, на который наведен курсор. Кроме того, в основном меню имеется иконка  , которая позволяет получать информацию о любом объекте, расположенном в активном окне.

Как правило, сеанс работы с графическим редактором состоит из следуюших этапов:

1) создание нового файла;
2) назначение имени проекта;
3) выбор набора инструментов;
4) ввод логических элементов;
5) установка размера и параметров сетки привязки;
6) перемещение символов;
7) ввод входных/выходных контактов;
8) присвоение имён контактам;
9) соединение символов;
10) присвоение имён цепям и шинам;
11) сохранение и проверка файла на наличие ошибок;
12) создание символа по умолчанию;
13) закрытие файла.

Создание нового файла

Выбрать команду ‘New’ (File menu). Появляется диалоговое окно, в котором необходимо выбрать тип файла - ‘Graphic Editor File’.

Выбрать тип расширения файла в выпадающем меню - .gdf. Выбрать Ok.

Появляется окно графического редактора без имени - ‘untitled’.

Необходимо сохранить файл - команда ‘Save as’.

Назначение имени проекта

В системе MAX PLUS II необходимо назначать рабочий файл как текущий проект перед компиляцией, моделированием и другими процессами. Система может работать только с одним проектом в данный момент времени. В проект включаются все файлы, имеющие одинаковое имя. Разработчик должен создавать отдельную директорию для каждого нового проекта:

- выбрать команду ‘Project Name’ (File menu). Появляется диалоговое окно;
- задать созданный *.gdf - файл как верхний уровень проекта;
- выбрать ‘Ok’.

В верхней части окна графического редактора появляется имя проекта.

В качестве альтернативы команде ‘Project Name’ можно выбрать команду ‘Project Set Project to Current File’ (File menu).

Выбор набора инструментов

В левой части окна графического редактора имеется меню, содержащее набор доступных инструментов ‘tool‘:

- selection tool (инструмент выбора);

- text tool (текст);

- ortogonal line tool (рисование ортогональных линий);
- diagonal line tool (рисование диагональных линий);
- arc tool (рисование дуг);
- circle tool (рисование окружностей).

Текущий инструмент выбирается левой кнопкой мыши. При загрузке редактора по умолчанию выбирается ‘selection tool‘ - инструмент, предназначенный для выбора объектов в окне и соединения контактов элементов проводниками (имеет форму стрелки).

Ввод логических элементов

В системе MAX PLUS имеются библиотеки, содержащие символы различных логических функций - примитивные - And, Or, Xor, Not и т.д., мегафункции и макрофункции.

Навести курсор (должен быть в форме стрелки) на свободный участок в окне графического редактора и нажать левую кнопку мыши - для определения точки привязки элемента. При этом рядом с точкой появляется выпадающее меню (Symbol menu). В Symbol menu необходимо выбрать команду ‘Enter Symbol‘. На экране появляется диалоговое окно ‘Enter Symbol‘ (рис. 22).

Данное окно позволяет найти библиотечный символ по имени, просмотреть содержимое всех подключенных библиотек, сменить диск и выбрать директорию расположения библиотек.

Например, введем ‘8count‘ в поле ‘Symbol Name‘. Выберем ‘Ok‘. Символ ‘8count‘ размещается верхним левым углом в точке привязки. Макрофункция ‘8count‘ - это 8-битный двоичный счётчик.

Повторить вышеописанные шаги для ввода других символов.

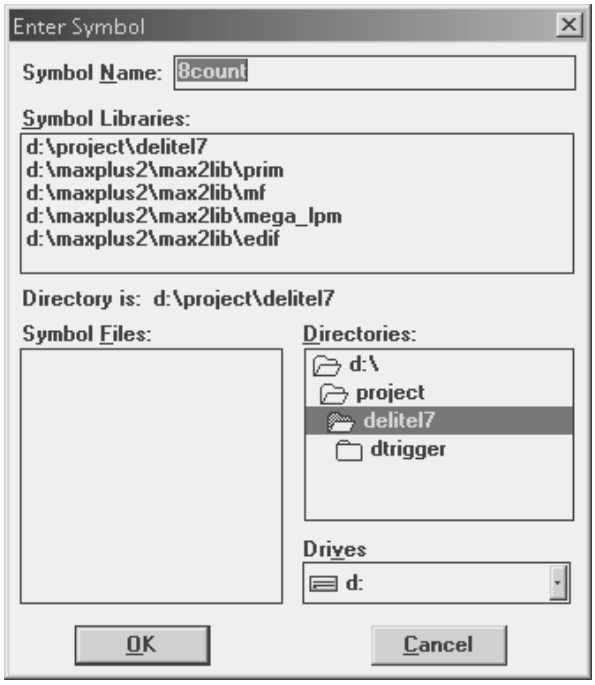


Рис.22. Меню ‘Enter Symbol‘

Установка сетки привязки

Для повышения наглядности и удобства работы с графическим редактором рекомендуется установить сетку привязки символов.
- Выбрать ‘Guideline Spacing’ из меню ‘Options’. Появляется диалоговое окно.
- Напечатать, например 15, во вкладах ‘X (Horizontal) Spacing’ и ‘Y (Vertical) Spacing’.
- Выбрать ‘Ok’.
- Выбрать ‘Show Guidelines’ из меню ‘Options’.

В результате в окне графического редактора появляется сетка с размером ячейки 15 единиц по горизонтали и вертикали.

Перемещение символов

- Выбрать ‘Selection tool’ из меню, расположенного в левой части экрана.
- Навести курсор на символ.
- Нажать левую кнопку мыши, и не отпуская её, переместить символ в требуемую позицию.

Таким образом, в графическом и символьном редакторах можно перемещать любые символы, графику, текстовые блоки.

Ввод входных/выходных контактов

- Дважды нажать левую кнопку мыши в свободной части экрана, при этом появляется диалоговое окно ‘Enter Symbol’.
- Напечатать ‘input’ или ‘output’ в поле ‘Symbol Name‘ и выбрать ‘Ok’.

В окне появляется требуемый символ. Если схема имеет несколько входов/выходов, необходимо произвести следующие операции.
- навести курсор на символ;
- нажать клавишу ‘Ctrl’, левую кнопку мыши и, не отпуская их, перемещать элемент вниз;

- отпустить левую кнопку мыши. При этом произошло копирование элемента. Для продолжения копирования нужно снова нажать левую кнопку мыши.
Данный способ позволяет копировать символы без буфера обмена.

Каждый введенный символ имеет идентификационный номер, расположенный в левом нижнем углу границ символа, обозначенных пунктиром. Номер соответствует порядку, в котором вводились символы.
Кроме того, каждому символу по умолчанию присваивается имя, обозначенное как ‘PIN_NAME’.

Присвоение имён контактам

Каждый входной/выходной контакт схемы должен иметь своё, отличное от других, имя.

Навести курсор на поле ‘PIN_NAME’ символа и дважды нажать левую кнопку мыши.

Напечатать требуемое имя. Если после этого нажать ‘Ввод’, то автоматически выберется контакт, расположенный ниже для редактирования имени.

Соединение символов

Перед соединением символов (элементов) их нужно, по возможности, разместить так, чтобы соединяемые контакты находились на одном уровне.
- Выбрать тонкую непрерывную линию в подменю ‘Line Style’ (Options menu). Эта линия рекомендуется для построения проводников.
- Нажать иконку ‘ortogonal line tool’ в левой части экрана. При этом курсор приобретает форму крестика.
- Навести курсор на контакт элемента и нажать левую кнопку мыши.
- Не отпуская кнопку, перемещать мышь до следующего контакта или излома проводника.

Для удаления линии нужно пометить её нажатием левой кнопки мыши и нажать клавишу ‘Del’ или ‘Backspace’.

Для рисования шин нужно выбрать толстую непрерывную линию в подменю ‘Line Style’ (Options menu).

Присвоение имён цепям и шинам

Сменить размер шрифта (выбрать Arial –10).

С помощью двойного нажатия мыши пометить требуемую цепь. При этом под цепью появляется маленькая квадратная точка, показывающая точку привязки имени.

Ввести имя.

Если введенное имя накладывается на символ, его можно переместить с помощью мыши.

Сохранение и проверка файла на наличие ошибок

Для того, чтобы проверить логическую корректность введенной схемы, нужно сохранить файл и проверить на наличие ошибок.

Выбрать ‘Project Save & Check’ из меню ‘File’. При этом файл сохраняется и на экране появляется окно компилятора (Compiler window). Модуль ‘Compiler Netlist Extractor’ проверяет файл и выдаёт сообщения об ошибках.

Создание символа по умолчанию

Имеется возможность создания символа (файл *.sym), соответствующего введенной схеме. Символ может быть использован в других схемах.

Выбрать ‘Create Default Symbol’ из меню ‘File’. Если символ для файла уже существует, выдаётся запрос о возможности перезаписи.

Закрытие файла

- Выбрать ‘Close’ из меню ‘File’.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Получить у преподавателя схему электрическую принципиальную и перечень элементов (если нужно).
2. Ознакомиться с работой схемы.
3. Определить элементы, которые не могут быть включены в проект.
4. Определить элементы, которых нет в библиотеках. Найти аналоги для замены элементов или создать новые элементы.
5. Создать электрическую схему устройства с помощью графического редактора системы MAX PLUS II.
6. Проверить схему на логическую корректность и наличие ошибок.

СОДЕРЖАНИЕ ОТЧЁТА

Отчет о работе должен содержать:

1. Протокол работы с графическим редактором.
2. Схему электрическую принципиальную устройства, полученную у преподавателя.
3. Описание работы проектируемого устройства.
4. Схему электрическую принципиальную устройства, полученную после преобразования для использования в системе MAX PLUS II.
5. Выводы о классах устройств и задач, поддающихся реализации на ПЛИС.
6. Выводы о возможностях систем разработки ПЛИС.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем различия процессов проектирования ПЛИС и других БИС программируемой логики?
2. Каковы основные способы описания проекта на ПЛИС? Их достоинства и недостатки.
3. Какие преимущества получает разработчик при использовании языков высокого уровня?
4. Графический редактор системы MAX PLUS II: его сильные и слабые стороны.
5. Какие классы схем могут быть реализованы на ПЛИС? Предъявляет ли САПР ПЛИС какие-либо особые требования к схемотехнике?

ПРИМЕР

В качестве примера рассмотрим описание делителя на четыре, собранного на микросхеме КР1533ИЕ5 (аналог SN74LS93). Микросхема КР1533ИЕ5 представляет собой четырехразрядный двоичный счетчик и содержит четыре триггера, срабатывающих по отрицательному фронту на информационных входах, а также дополнительные связи, реализующие в микросхеме две секции: счетчик-делитель на два и трехразрядный счетчик-делитель на восемь. Каждая секция используется отдельно, а для получения 4-разрядного счетчика используется внешняя связь выхода счетчика-делителя на два со входом трехразрядного счетчика. Счетчик имеет один вход, на который подается последовательность бит (бит-вектор), и один выход, который может быть представлен в виде шины.

Открываем систему MAX PLUS II и с помощью команды ‘New’ (File menu) создаем новый файл.

С помощью команды ‘Choose Project Name’ (File menu) определим имя проекта – ‘Counter’.

Для ввода логических элементов двойным нажатием мыши определяем точку привязки, при этом появляется диалоговое окно ‘Enter Symbol‘. В поле ‘Symbol name’ нужно указать указать имя вводимого символа. В нашем случае – ‘7493’. Кроме того, потребуется ввести вход (Input), выход (Output) и специальный символ Gnd, используемый для подключения контактов к общей шине. После выполнения этих действий в окне появляются символы (рис.23).

Размещаем элементы с помощью мыши и соединяем одним из доступных инструментов. Счетчик имеет четыре выхода, их целесообразно организовать в виде шины. Рисование шины происходит следующим образом. Двойным нажатием мыши выделяем поле ‘PIN_NAME’ элемента OUTPUT, в котором задаем разрядность шины. В нашем случае - Out [3..0]. Далее выбираем инструмент для рисования линий (ortogonal/diagonal line) и задаем тип линии - толстую непрерывную (меню ‘Line Style’ - Options menu). Рисуем участок шины, присоединенный к элементу OUTPUT. Для рисования проводников изменяем тип линии на тонкую непрерывную. Обозначаем проводники, входящие в шину, как показано на рис.24.

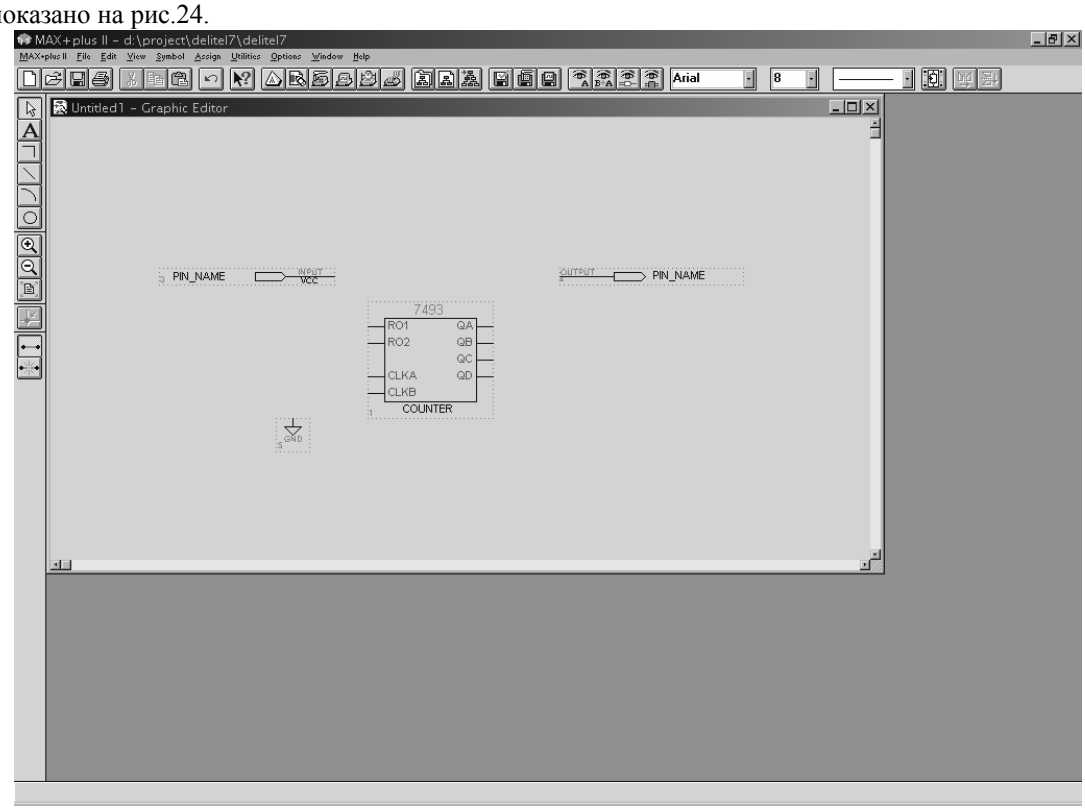


Рис.23. Файл ‘Counter’ с введенными символами

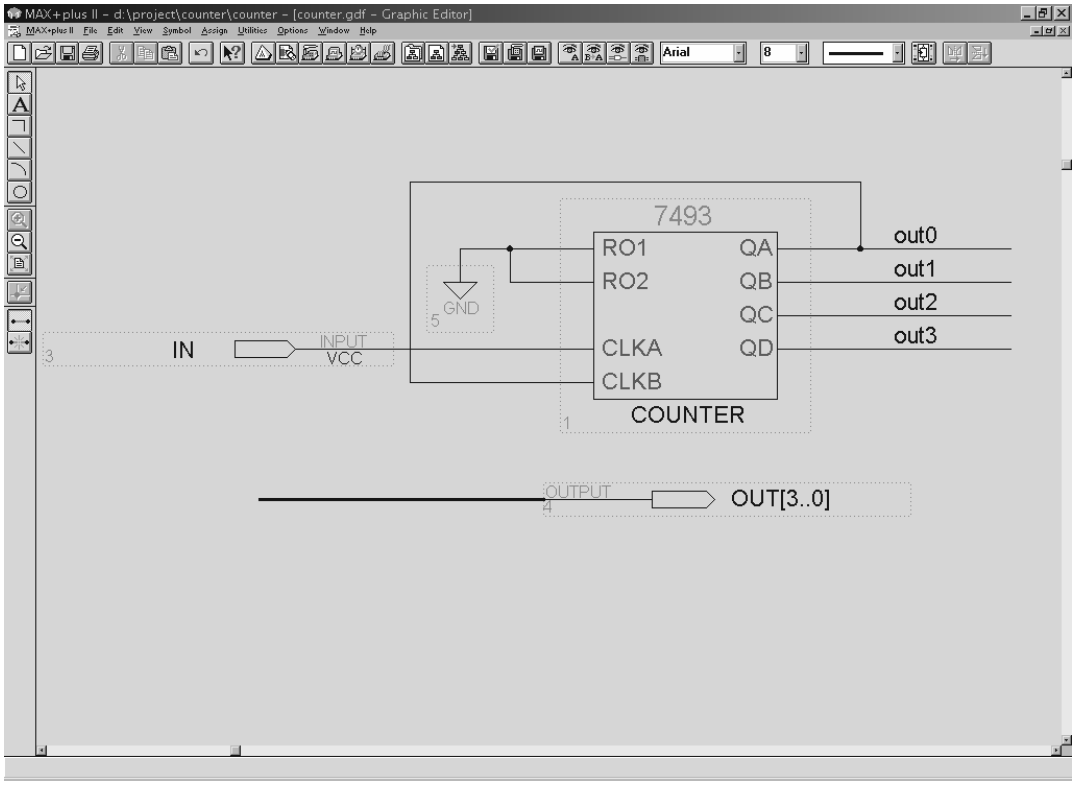


Рис.24. Файл ‘Counter’ с соединением элементов

Проверим логическую корректность введенной схемы с помощью команды ‘Project Save & Check’ из меню ‘File’. Модуль ‘Compiler Netlist Extractor’ выдаёт сообщение ‘0 errors & 0 warnings’. Введенная схема логически корректна, можно переходить к следующему этапу разработки.

4.2. Лабораторная работа №2. Работа с компилятором системы MAX PLUS II

Цель работы: изучение процесса синтеза проекта на ПЛИС, получение навыков работы с компилятором системы MAX PLUS II.

ВВЕДЕНИЕ

Современные САПР ПЛИС обеспечивают сквозное проектирование с полной автоматизацией всех этапов разработки. Проекты с высокоплотной программируемой логикой предъявляют исключительные требования к таким системам: набор инструментальных средств CAM-систем должен обеспечивать простоту проектирования в сочетании с высокой производительностью.

Методология проектирования ПЛИС представлена на рис.25 и состоит из взаимосвязанных шагов: входного описания проекта, синтеза и проверки характеристик полученного устройства. Процесс проектирования

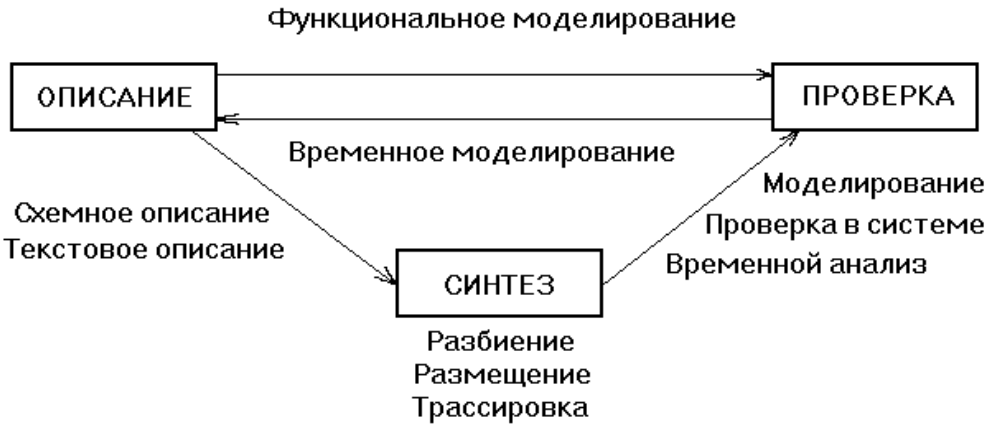


Рис.25. Маршрут проектирования ПЛИС

носит итеративный характер, с обратными связями для коррекции и оптимизации полученных результатов.

Наиболее популярными методами входного описания являются схемное и текстовое. FPGA/CPLD библиотеки символов доступны для схемных редакторов таких фирм, как Viewlogic, OrCAD, Mentor Graphics, Cadence и др. Эти библиотеки отражают широкий спектр логических функций, которые могут быть реализованы на ПЛИС, кроме того, поддерживается экспорт/импорт схем в формате EDIF.

Поведенческие методы входного описания (текстовое, булевы уравнения, конечные автоматы) поддерживаются такими производителями CAE-систем, как Data I/O, Logical Devices, MINC, ISDATA, Viewlogic, Exemplar и др.

При сложности проекта выше 10 000 вентилей низкоуровневые описания (например схемное) становятся громоздкими и неудобными, а высокоуровневые поведенческие описания позволяют значительно повысить производительность труда разработчика.

После того как проект был формализован одним из методов, САПР приступает к реализации (синтезу) логики в ресурсах заданной архитектуры ПЛИС.

Синтез проекта – самый ответственный этап проектирования ПЛИС, от которого во многом зависят характеристики будущего устройства (количество кристаллов, площадь кристалла, быстродействие, энергопотребление, стоимость и др.). В системе MAX PLUS II синтез проекта осуществляет отдельный программный модуль – компилятор (Compiler). На рис.26 представлена структура компилятора системы MAX PLUS II.

Обрабатывая проект, компилятор считывает входной файл и создаёт файлы для программирования, моделирования и временного анализа.

Исходная информация обрабатывается модулем ‘Compiler Netlist Extractor’, который создаёт один или несколько двоичных файлов списков цепей (Compiler Netlist Files .cnf).

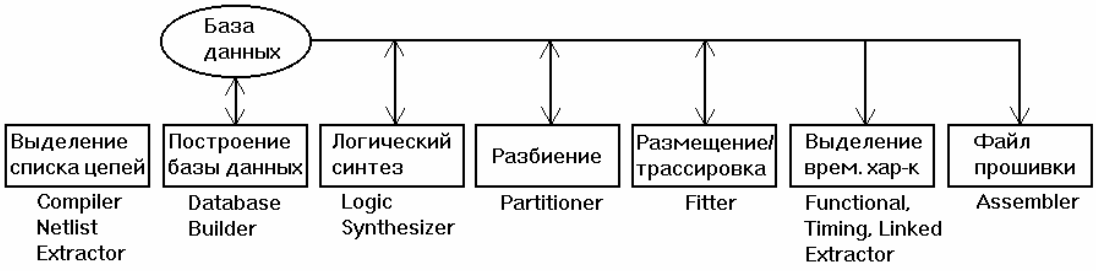


Рис.26. Структура компилятора

Дополнительно модуль создаёт файл иерархии проекта (Hierarchy Interconnect File .hif). Файл иерархии связывает все файлы проекта.

Модуль ‘Database Builder’ использует файл иерархии, чтобы связать файлы списков цепей в один проект: таким образом сохраняются электрические связи проекта. Кроме того, проверяется логическая полнота описания, подсоединение цепей к входным/выходным контактам, логические ошибки (оборванные цепи и т.д.). Каждый модуль компилятора обрабатывает и дополняет базу данных проекта.

Полученный список цепей обрабатывается модулем логического синтеза (Logic Syntesizer) и преобразуется в систему булевых уравнений. На этапе логического синтеза происходит также минимизация логических функций, удаление излишней и неиспользуемой логики. Задача логического синтеза состоит в том, чтобы использовать логические ресурсы микросхемы (прибора) настолько эффективно, насколько это возможно для данной архитектуры. Опции логического синтезатора позволяют разработчику влиять на результат логического синтеза и получать максимальные преимущества от выбранной архитектуры прибора.

Далее, проект в виде системы логических функций обрабатывается модулем ‘Partitioner’. Модуль оценивает сложность проекта по количеству используемых входных/выходных контактов и эквивалентных вентилей. Если проект слишком велик для реализации на одном кристалле, ‘Partitioner’ производит разбиение проекта на несколько кристаллов, пытаясь минимизировать количество цепей, необходимых для их соединения. Разбиение может производиться в автоматическом или ручном режимах.

Модуль ‘Fitter’ отвечает за размещение и трассировку проекта. Модуль использует набор эвристических правил для выбора лучшей возможной реализации проекта на одном или нескольких кристаллах. В процессе работы модуля формируется файл отчёта (Report File) с указанием используемых ресурсов кристалла. Результаты работы можно просмотреть с помощью редактора разводки (Floorplan Editor).

Модуль ‘Functional Extractor’ создаёт файл для функционального моделирования (Simulator Netlist File .snf). Компилятор создаёт этот файл без синтеза кристалла (в нем отсутствует информация о временных характеристиках). Функциональное моделирование используется для проверки правильности функционирования устройства на ранних стадиях разработки.

Модуль ‘Timing Extractor’ создаёт файл для временного моделирования и анализа (.snf), в котором содержится информация о временных характеристиках кристалла.

Модуль ‘Linked Extractor’ создаёт файл (.snf), в котором содержится информация для функционального и временного моделирования. Такие файлы используются для совместного моделирования нескольких проектов, состоящих из кристаллов разных семейств.

Модуль ‘Assembler’ преобразует топологию кристалла в один или несколько файлов прошивки:

- Programmer object files (.pof),
- SRAM object files (.sof),
- JEDEC files (.jed),
- Hexadecimal (Intel-format) files (.hex),
- Tabular text files (.ttf).

ПОДГОТОВКА К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Для компиляции проекта нужно выполнить следующую последовательность шагов:

1) открыть окно компилятора;
2) выбрать семейство микросхем;
3) выбрать опцию ‘Smart Recompile’;
4) включить утилиту ‘Design Doctor’;
5) установить опцию ‘Security Bit’;
6) выбрать опцию ‘Global Project Logic Syntesis’;
7) включить утилиту ‘Timing SNF Extractor’;
8) определить информацию для файла отчёта;
9) запустить компилятор;
10) определить и исправить ошибки;
11) получить рекомендации об исправлении ошибок;
12) изучить файл отчёта.

Открытие окна компилятора

Выбрать опцию ‘Compiler’ (MAX+PLUS II menu). На экране появляется окно компилятора (рис.27).

Количество модулей в окне зависит от того, какие установки были сделаны в основном меню перед запуском компилятора. Опции меню ‘Processing’ и ‘Interfaces’ содержат команды для настройки всех модулей.

Выбор семейства микросхем

Разработчик может выбрать любое семейство микросхем, поддерживаемых системой MAX PLUS II. Компилятор может выбрать тип микросхемы самостоятельно.

Выбрать опцию ‘Device’ (Assign menu). Появляется диалоговое окно (рис.28) .

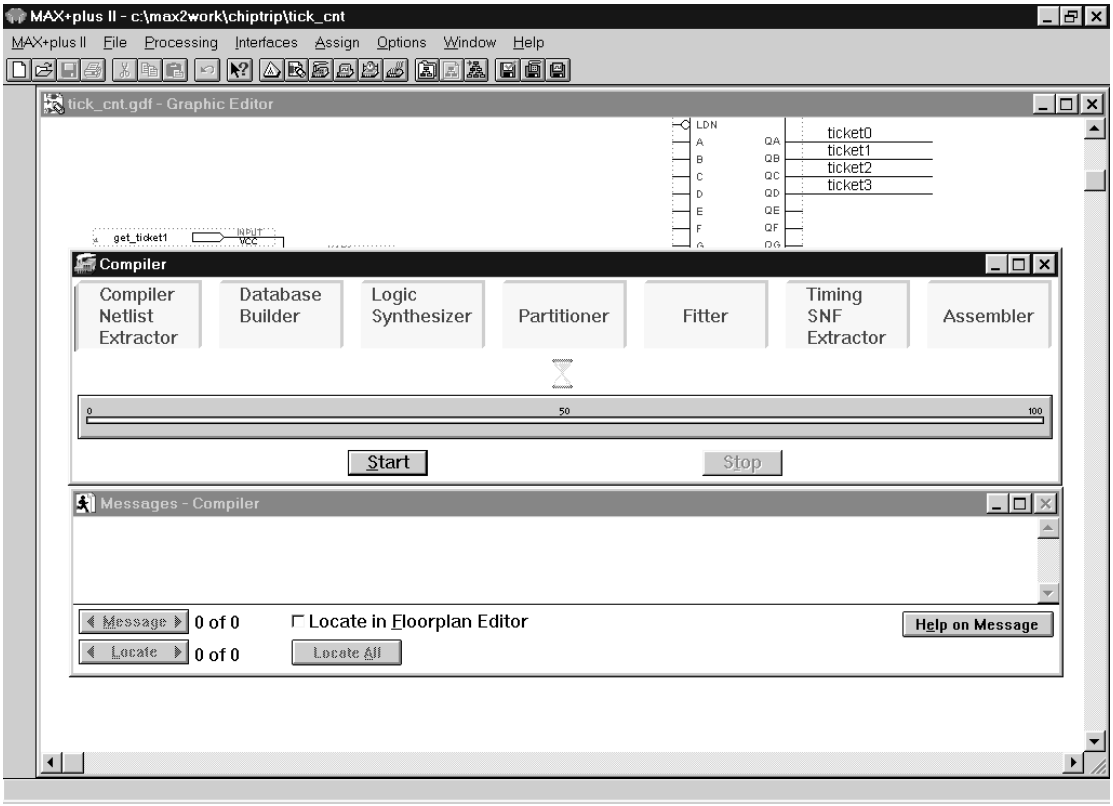


Рис.27. Окно компилятора

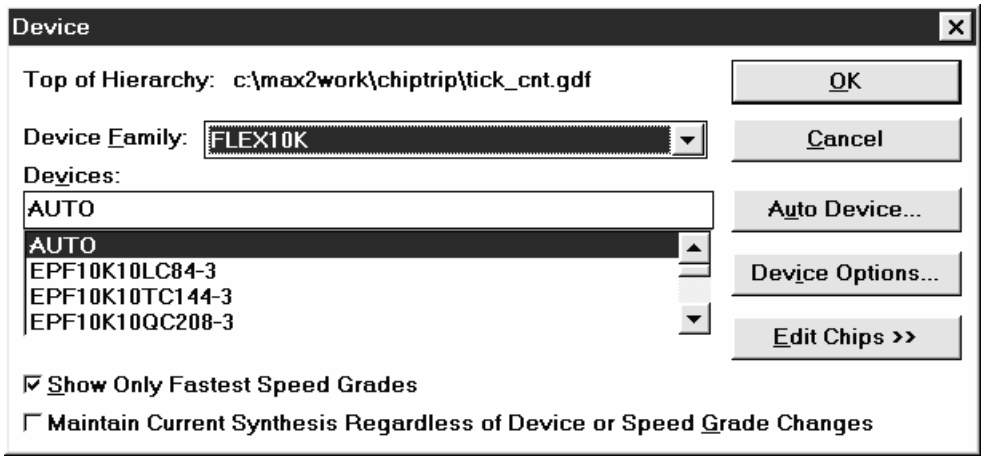


Рис.28. Окно выбора семейства микросхем

Выбрать семейство микросхемы из выпадающего окна ‘Device Family’.

- Выбрать тип микросхемы в окне ‘Devices’. Если необходимо, выбрать ‘AUTO’ (автоматический выбор).

Нажать ‘Ok’.

Опция ‘Smart Recompile’

Если Опция ‘Smart Recompile’ включена, компилятор сохраняет в базе данных информацию о проекте для использования в последующих компиляциях. При перекомпиляции с этой опцией компилятор определяет, какие части проекта не нужно обрабатывать заново.

Выбрать опцию ‘Smart Recompile’ (Processing menu) (рис.29).

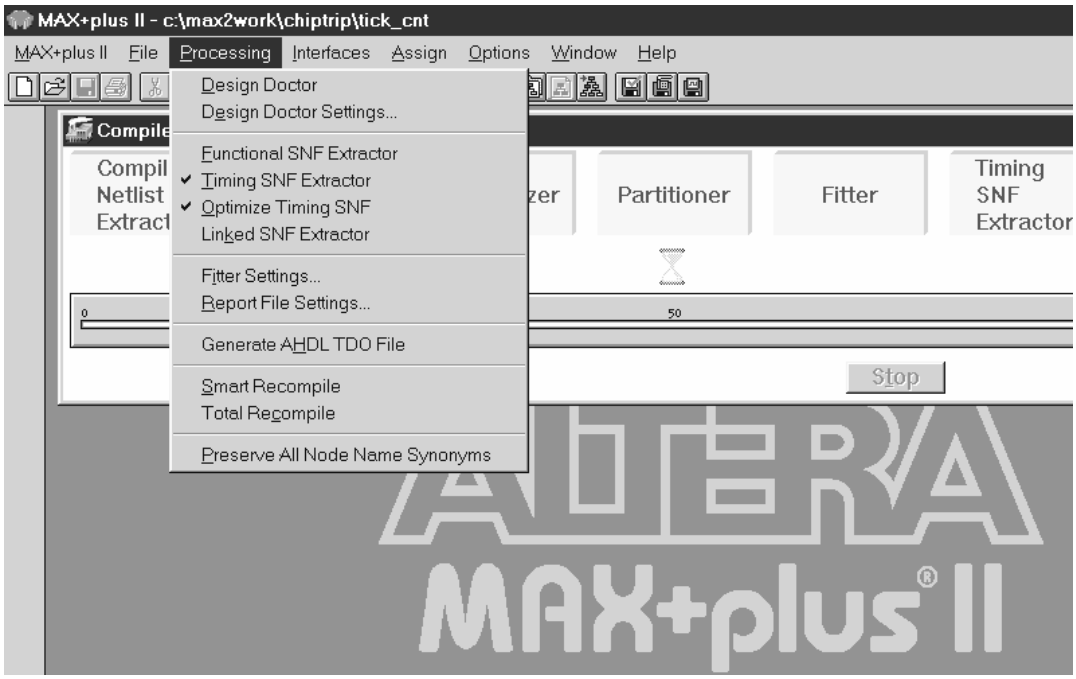


Рис.29. Processing menu

Утилита ‘Design Doctor’

В процессе компиляции утилита ‘Design Doctor’ проверяет все файлы проекта на наличие ошибок, которые могут повлиять на надёжность разрабатываемого устройства. Проверяется система синхронизации проекта, времена установки/сброса, статические состязания и т.д. Проверка производится на основе набора правил, который может быть выбран разработчиком.

- выбрать опцию ‘Design Doctor’ (Processing menu);
- выбрать набор правил в подменю ‘Design Doctor Settings’ (Processing menu) (рис.30);
- нажать ‘Ok’.

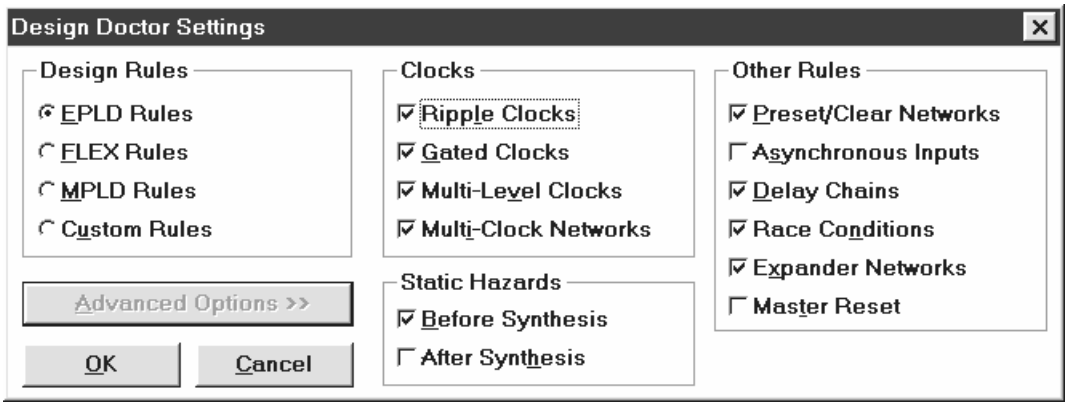


Рис.30. Подменю ‘Design Doctor Settings’

Опция ‘Security Bit’

Установка бита секретности позволяет закрыть микросхему от несанкционированного считывания. Бит секретности устанавливается только для приборов с энергонезависимой прошивкой (семейства Classic, MAX).

Выбрать опцию ‘Global Project Device Options’ (Assign menu). Появляется диалоговое окно (рис.31).

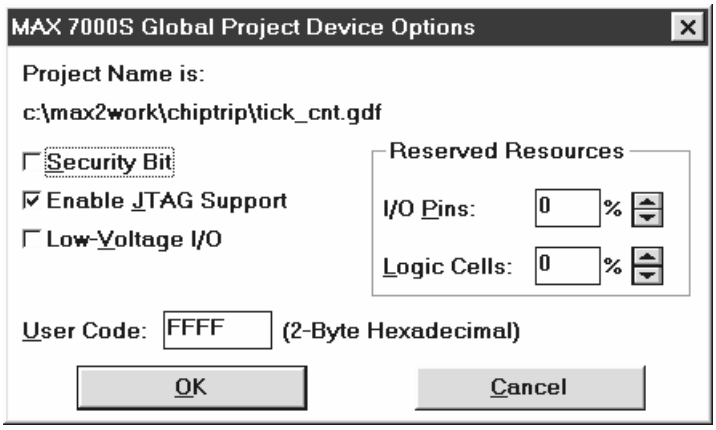


Рис.31. Опция ‘Global Project Device Options’

Включить (если нужно) бит секретности.
Нажать ‘Ok’.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Получить у преподавателя задание на исследование определённых опций и команд компилятора.
2. Произвести компиляцию проекта.
3. Сделать выводы о возможностях и недостатках компилятора.

СОДЕРЖАНИЕ ОТЧЁТА

1. Краткие теоретические сведения о структуре компилятора.
2. Описание основных опций и команд компилятора.
3. Протокол работы с компилятором.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каковы основные этапы маршрута проектирования ПЛИС?
2. Структура компилятора ПЛИС.
3. Какие особенности вносит в работу компилятора использование поведенческих методов входного описания?
4. Что такое «кремниевый» компилятор? Области применения, преимущества и недостатки.
5. Каким образом происходит передача текстовых описаний проекта между различными САПР?
6. Что является результатом успешной компиляции проекта в САПР ПЛИС?

ПРИМЕР

1. Выбираем опцию ‘Compiler’ (MAX+PLUS II menu). На экране появляется окно компилятора.
2. Выбираем опцию ‘Device’ (Assign menu). В диалоговом окне назначаем семейство микросхем – FLEX 10K и тип микросхемы – FLEX10K10LC84-3.
3. Выбираем опцию ‘Global Project Device Options’ (Assign menu). Появляется диалоговое окно (рис.32), оно отличается от окна, приведенного на рис.31. В частности отсутствует установка бита секретности, поскольку семейство FLEX 10K имеет энергозависимую прошивку. Наибольший интерес представляет вкладка ‘Configuration scheme’, в ней задается режим конфигурации микросхемы. В данном случае имеются три варианта: ‘Passive Serial’, ‘Passive Parrallel Synchronous’, ‘Passive Parrallel Asynchronous’. Подробно эти режимы описаны в работе №5 ‘Программирование ПЛИС’. В нашем случае нужно выбрать режим ‘Passive Serial’, поскольку он не требует применения дополнительных внешних устройств (контроллеров, ПЗУ и т.д.). Прошивка кристалла может осуществляться с помощью загрузочного кабеля (ByteBlaster) непосредственно с компьютера.
4. Выбираем опцию ‘Pin/Location/Chip’ (рис.33). Опция позволяет производить привязку внешних выводов проекта к цоколевке конкретной микросхемы – данная возможность чрезвычайно полезна на практике, поскольку, как правило, разработка печатной платы и кристалла ПЛИС ведутся параллельно.

Для использования опции ‘Pin/Location/Chip’ разработчик должен иметь таблицу назначения выводов кристалла. Как правило, подобная информация распространяется фирмами-разработчиками в ежегодных выпусках ‘Data Book’. В приложении приведена информация о назначении выводов ПЛИС ф.Altera и, в частности, для микросхем серии FLEX 10K.

5) Выбираем опции, описанные выше, если они нужны, и компилируем проект.

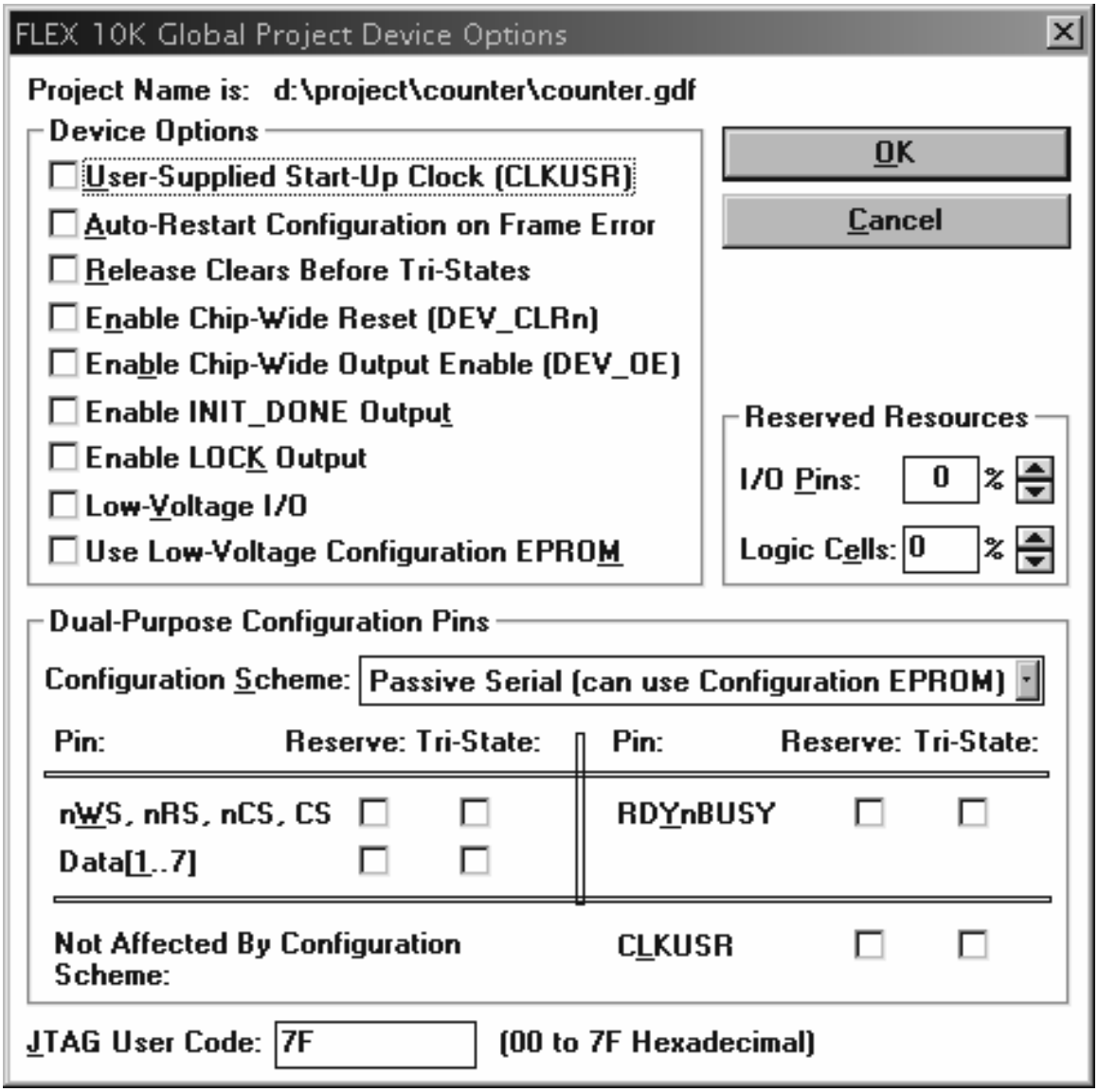


Рис.32. Опция ‘Global Project Device Options’ для FLEX 10K

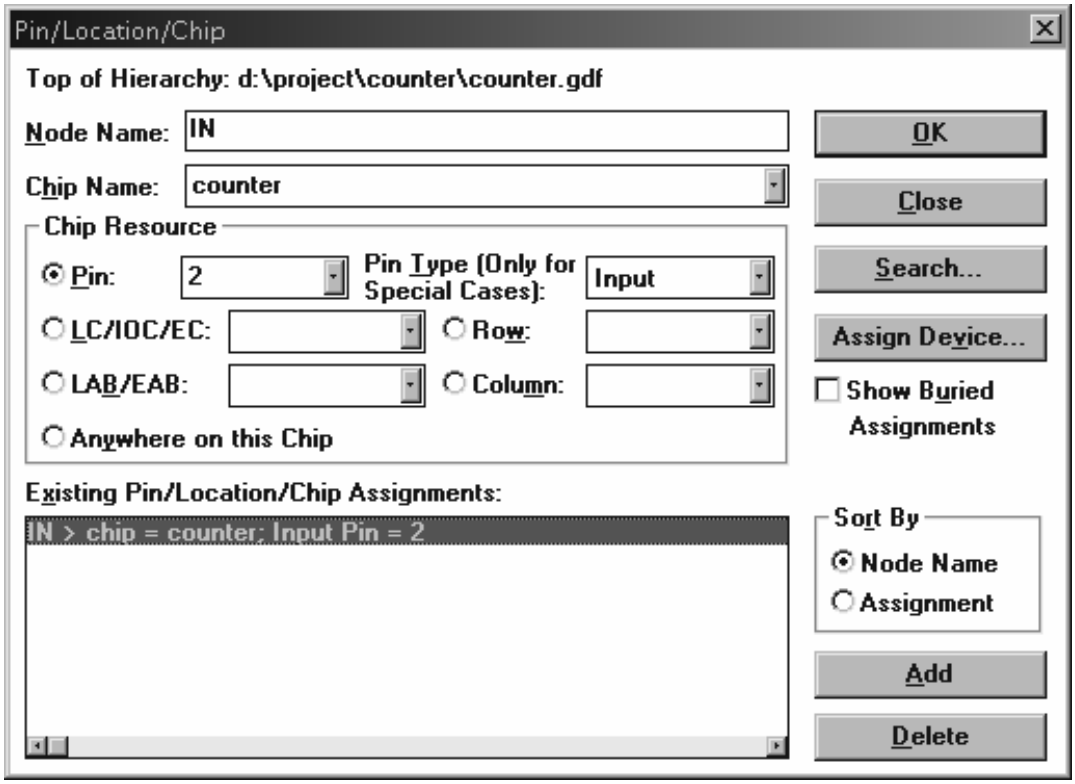


Рис.33. Опция ‘Pin/Location/Chip’

4.3. Лабораторная работа № 3. Логический синтез проекта на ПЛИС

Цель работы: изучение процесса логического синтеза проекта на ПЛИС, получение навыков работы с компилятором системы MAX PLUS II.

ВВЕДЕНИЕ

Логический синтез позволяет выбирать решения, оптимальные по производительности, числу кристаллов и стоимости, а также имеет тесную связь с аппаратурой программирования микросхем. Первый этап синтеза приводит к описанию проекта на уровне логических функций.

Полученный список цепей обрабатывается модулем логического синтеза (Logic Syntesizer) и преобразуется в систему булевых уравнений. На этапе логического синтеза происходит также минимизация логических функций, удаление излишней и неиспользуемой логики. Задача логического синтеза состоит в том, чтобы использовать логические ресурсы микросхемы (прибора) настолько эффективно, насколько это возможно для данной архитектуры. Опции логического синтезатора позволяют разработчику влиять на результат логического синтеза и получать максимальные преимущества от выбранной архитектуры прибора.

ПОДГОТОВКА К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Опция ‘Global Project Logic Syntesis’

С помощью опции ‘Global Project Logic Syntesis’ разработчик может выбрать стиль логического синтеза, который будет определять работу модуля ‘Logic Syntesizer’. По умолчанию, для нового проекта устанавливается стиль ‘Normal’. Установка логических опций стиля позволяет оптимизировать проект по критериям занимаемой площади на кристалле и быстродействию (рис.34).

Опция меню ‘Multi-Level Syntesis for MAX Devices’ – многоуровневый синтез - позволяет получить максимальные преимущества от всех доступных логических опций, включая все опции из

меню ‘Define Synthesis Style’ и ‘Advanced Options’ (рис.35,36). Если опция выключена, используется стандартный синтез.

Опция меню ‘One-Hot State Machine Encoding’ – автоматически применяет ‘One-Hot’ – кодирование к проекту (так называемое соседнее кодирование, при котором в каждом соседнем коде изменяется только один разряд). Не рекомендуется совмещать ручное кодирование с автоматическим. Следует помнить, что соседним кодированием нельзя закодировать циклы нечётной длины, соседние состояния, имеющие три и более перехода.

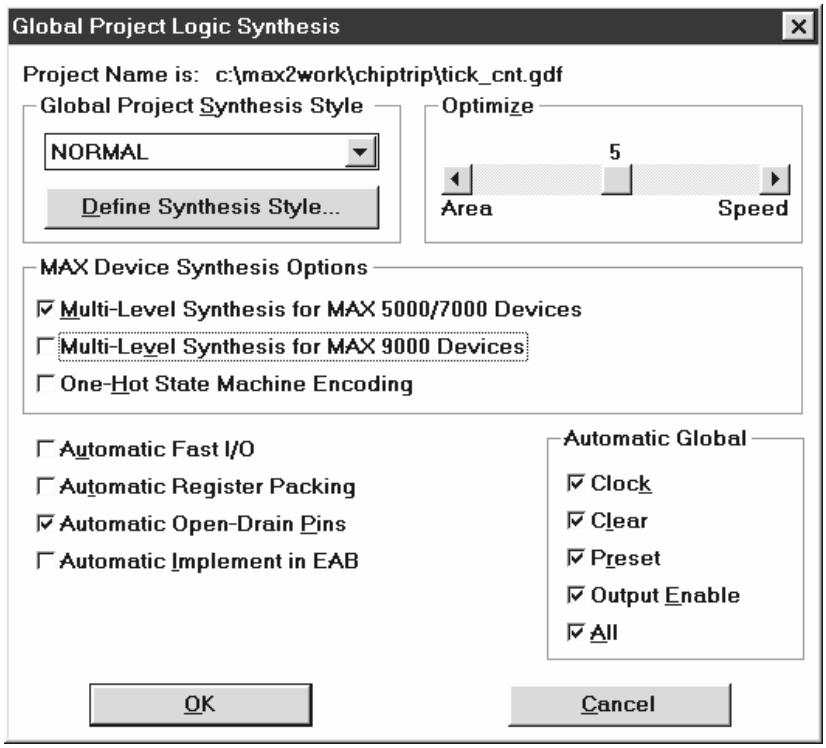


Рис.34. Меню ‘Global Project Logic Syntesis’

Опция ‘Automatic Fast I/O’ – позволяет компилятору размещать регистры и комбинационную логику непосредственно в блоках ввода/вывода, тем самым улучшая временные характеристики кристалла.

Опция ‘Automatic Register Packing’ – доступна только для микросхем семейств MAX9000 и FLEX10K; позволяет размещать в одной логической ячейке комбинаторную и регистровую логику, имеющую общий вход данных.

Опция ‘Automatic Open-Drain Pins’ – позволяет компилятору назначать выводы с открытым коллектором. Тот же эффект достигается введением в проект примитива ‘OPNDRN’.

Опция ‘Automatic Implement in EAB’ – используется только при реализации комбинаторных макрофункций в блоках EAB микросхем семейства FLEX10K.

Меню ‘Automatic Global’ – позволяет компилятору производить контроль использования ресурсов. Если несколько триггеров управляются одним сигналом, он назначается как глобальный (Clock, Clear, Preset – в зависимости от выполняемой функции) и реализуется на специально выделенной шине кристалла с высокостабильными временными параметрами. Если несколько тристабильных буферов управляются одним сигналом, он назначается как глобальный - ‘Output Enable’.

Простейший путь установки логических опций – использование готовых стилей синтеза. Система MAX PLUS II имеет три встроенных стиля синтеза: Normal, Fast и WYSIWYG. Разработчик может создать свой собственный стиль разработки (рис.35).

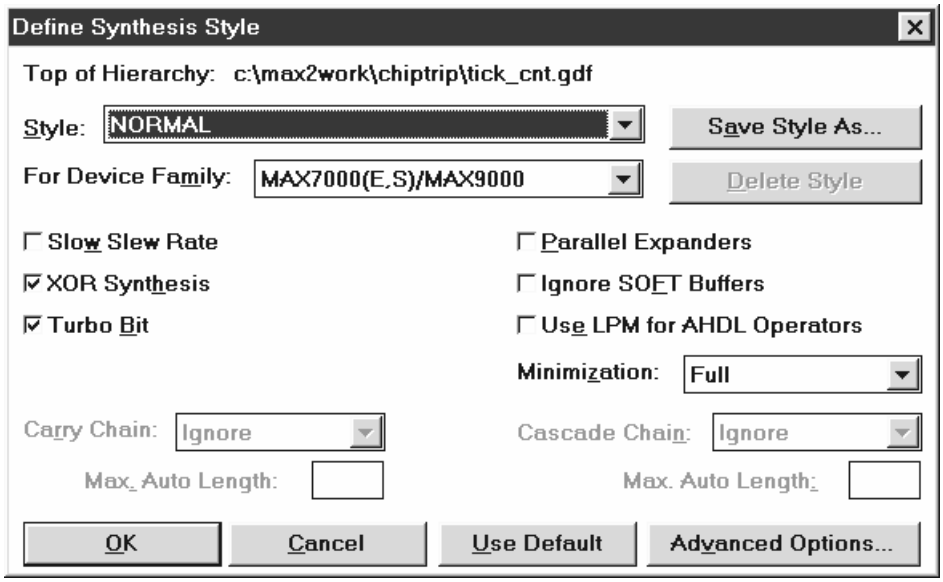


Рис.35. Подменю ‘Define Synthesis Style’

Опция ‘Slow Slew Rate’. При выборе этой опции на выходных и двунаправленных контактах переходы 0-1 и 1-0 формируются более медленно (затягиваются фронты переходов), что приводит к снижению `шума’ схемы.

Опция ‘XOR Syntesis’ – это опция логической минимизации, наиболее эффективна, когда в проекте используются макрофункции и мегафункции. При включении этой опции модуль ‘Logic Synthesizer’ минимизирует логические функции путём включения дополнительных вентилей XOR для соединения комбинаторных логических ячеек. Данная опция предназначена для архитектур логических ячеек семейств MAX5000, MAX7000, MAX9000.

Опция ‘Turbo Bit’ – контрольный бит для выбора характеристик кристалла. Если опция включена, будут улучшены временные характеристики (быстродействие), но возрастёт потребляемая мощность.

Опция ‘Parallel Expanders’ – позволяет использовать параллельные термы расширения соседних ячеек. В результате снижается количество дополнительных термов расширения, повышается быстродействие, однако могут возникнуть сложности при трассировке кристалла.

Опция ‘Ignore Soft Buffers’ – позволяет компилятору игнорировать Soft-буферы, добавленные вручную.

Опция ‘Use LPM …’ – используется только для AHDL-описаний.

Опция ‘Minimization’ – позволяет контролировать степень уменьшения и упрощения логики в процессе логического синтеза. Опция может быть установлена в одно из трёх значений ‘Full’, ‘Partial’ и ‘Default’. Установка ‘Full’ позволяет минимизировать логику настолько, насколько это возможно, удаляя излишние термы. Минимизация производится по правилу Де Моргана. Получаемая логика, как правило, значительно отличается от исходной. Установка ‘Partial’ позволяет выполнять простую оптимизацию логики, без исключения термов. Такая оптимизация подходит для реализации регистровых схем на микросхемах семейства MAX, а также, если в проекте имеются примитивы SOFT, LCELL, OUTPUT, OUTPUTC, BIDIR, BIDIRC, управляемые комбинаторной логикой.

Опции ‘Carry Chain’ и ‘Cascade Chain’ – используются при реализации макрофункций на кристаллах семейств FLEX6000, FLEX8000, FLEX10K для контроля использования цепей переноса.

Подменю ‘Define Synthesis Style’ имеет дополнительные опции, содержащиеся в подменю ‘Advanced Options’ (рис.36).

Опция ‘SOFT Buffer Insertion’ – директива модулю логического синтеза, позволяющая вставлять SOFT-буферы в выгодные места проекта. Если эта опция включена, нелегальные комбинаторные обратные связи автоматически фиксируются.

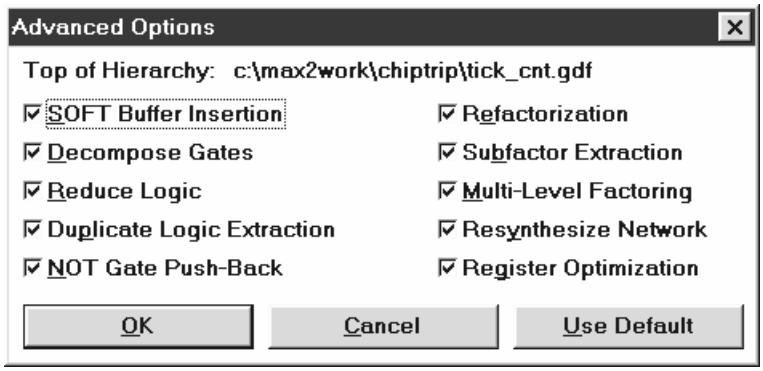


Рис.10. Подменю ‘Advanced Options’

Если опция выключена, все логические функции, кроме функций, связанных с введёнными вручную LSELL и SOFT – буферами приводятся к дизъюнктивной нормальной форме и должны быть включены в одну логическую ячейку.

Опция ‘Decompose Gates’ - директива модулю логического синтеза, позволяющая производить разбиение сложных логических функций, и объединять простые. Если опция выключена, логический синтез и другие логические опции могут не привести к эффективным результатам. Данная опция доступна только в том случае, если выбраны опции ‘Multi-Level Syntesis for MAX Devices’ и ‘SOFT Buffer Insertion’.

Опция ‘Reduce Logic’ (сокращение логики) - директива модулю логического синтеза, при которой минимизация функций проходит с учётом всего проекта. Производится удаление всех входов вентилей, соединенных с шинами ‘gnd’ и ‘+5v’. Если опция выключена, минимизация проводится локально, обрабатываются только ближайшие термы.

Следующий пример показывает, как может быть сокращена логика с помощью опции ‘Reduce Logic’:

до: t = a&(b # w); y = a # w&c; w = (z&a # !z&b); z = !c&d;

после: t = a&(b # z); y = a # b&c; z = !c&d;

где ! – операция отрицания, # - операция ИЛИ.

Опция ‘Duplicate Logic Extraction’ - директива модулю логического синтеза, при которой происходит удаление дублированных логических функций из проекта. Если два вентиля производят одинаковую логическую функцию, второй будет удалён из цепи, а первый вентиль будет разветвлён по выходу на нагрузку второго. Однако, если выход вентиля соединён с примитивом LCELL, сигнал на выходе воспринимается как уникальный и не обрабатывается данной опцией.

Следующий пример показывает, как может быть сокращена логика с помощью опции ‘Duplicate Logic Extraction’:

до: t = a&w; y = b&z; z = c&d # e&f; w = c&d # e&f;

после: t = a&z; y = b&z; z = c&d # e&f.

Опция ‘NOT Gate Push-Back’ – позволяет продвинуть инверсию обратно через регистр и реализовать её на D-входе регистра в наиболее выгодном для этого месте проекта. Эта опция доступна как для многоуровневого, так и для стандартного синтеза.

Опция ‘Refactorization’ – позволяет переносить часть логики на другие вентили проекта (операция, аналогичная вынесению за скобки). Эта опция может значительно уменьшить количество вентилей в проекте, но могут возрасти задержки прохождения сигналов в результате увеличения количества уровней логики. Если выход вентиля соединён с примитивом LCELL, сигнал на выходе воспринимается как уникальный и не обрабатывается данной опцией.

Опция ‘Subfactor Extraction’ – позволяет выделять подфункции, общие для двух или нескольких функций проекта и создавать новые соединения для реализации этих общих подфункций. Эта опция может значительно уменьшить количество вентилей в проекте, но могут возрасти задержки прохождения сигналов в результате увеличения количества уровней логики.

Следующий пример показывает, как может быть сокращена логика с помощью опции ‘Subfactor Extraction’:

до: t = a&b&(c#d)&e; y = (c#d)&e&(f#g);

после: t = a&b&z; y = z&(f#g); z = (c#d)&e.

Опция ‘Multi-Level Factoring’ – позволяет преобразовывать функции из дизъюнктивной нормальной формы в многоуровневую форму.

Пример:

до: t = a&e # b&c&e # b&d&e # a&f # b&c&f # b&d&f;

после: t = (a # b&(c # d))&(e # f).

Опция ‘Resynthesize Network’ – позволяет полностью перестроить структуру соединений проекта: сначала комбинаторные цепи частично или полностью приводятся к дизъюнктивной нормальной форме, а затем рефакторизуются. Логический синтезатор сравнивает полученное соединение с оригиналом и выбирает лучшее. Временные характеристики и форма полученной логики могут значительно отличаться от оригинальной, заданной в файле.

Пример:

до: t = e&(a # bc) # b((a#d)e # cf) # f&(a # b(c#d));

после: t = (a # b&(c#d))&(e#f).

Опция ‘Register Optimization’ – позволяет удалять дублированные регистры. Данная опция доступна только для многоуровневого синтеза.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Получить у преподавателя задание на исследование определённых опций меню ‘Global Project Logic Syntesis’.
2. Произвести компиляцию проекта с изменением опций логического синтеза.
3. Сравнить полученные результаты.
4. Сделать выводы о влиянии этапа логического синтеза на разрабатываемое устройство.

СОДЕРЖАНИЕ ОТЧЁТА

Краткие теоретические сведения о логическом синтезе.

Описание основных опций меню ‘Global Project Logic Syntesis’.

Протокол работы с компилятором.

Сравнительный анализ результатов, полученных при перекомпиляциях проекта с изменением опций логического синтеза.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие основные задачи решаются на этапе логического синтеза?
2. Методы кодирования состояний цифровых автоматов: достоинства и недостатки. Как используется кодирование в САПР MAX PLUS II?
3. Основные законы алгебры логики. Правило де Моргана.
4. Какие опции компилятора позволяют контролировать процесс логического синтеза?
5. Каким образом задается логика, не подлежащая минимизации? Для чего используются такие возможности компилятора?

4.4. Лабораторная работа №4. Моделирование и временной анализ проекта на ПЛИС

Цель работы: изучение теоретических основ моделирования и временного анализа цифровых устройств, методики моделирования проекта на ПЛИС, получение навыков работы с симулятором системы MAX PLUS II.

ВВЕДЕНИЕ

Моделирование цифрового устройства заключается в построении его математической модели – системы соотношений, описывающей поведение этого устройства с заданной точностью, и последующем анализе поведения этой модели по её реакции на входные воздействия.

С помощью моделирования в системе автоматизированного проектирования цифровых устройств решаются следующие задачи:

- проверка правильности логического функционирования устройства;
- проверка временных характеристик устройства;
- проверка функционирования цепей установки в начальное состояние;
- анализ состязаний и рисков сбоя;
- определение полноты тестов.

Существует много различных методов и алгоритмов моделирования. Основными характеристиками алгоритмов моделирования являются адекватность, быстродействие и объём памяти, необходимый при реализации. Под адекватностью понимается степень соответствия результатов моделирования истинному поведению исследуемого устройства. Для комбинационных схем алгоритмы моделирования гарантируют полную адекватность с установившимися значениями сигналов. Поведение последовательностных схем в общем случае неоднозначно из-за неопределенности начальных состояний и состязаний между сигналами, что усложняет задачу моделирования таких цифровых устройств. Неоднозначность учитывается в разной степени различными методами. Ещё труднее адекватно моделировать переходные процессы, однако для большинства практически важных задач это не требуется: достаточно правильно вычислять установившиеся значения сигналов.

Адекватность моделирования зависит в основном от принятой модели цифрового устройства, моделей элементов и сигналов, способа учёта временных соотношений между сигналами. Как правило, повышение степени адекватности связано со снижением быстродействия и увеличением требуемого объёма памяти, поскольку усложнение модели ведёт к увеличению её объёма и времени обработки.

Выбор метода моделирования часто является результатом компромисса между различными требованиями и зависит от класса цифрового устройства, вида решаемой задачи, имеющихся в распоряжении вычислительных ресурсов и т.д.

Для решения задачи анализа цифровых схем в настоящее время имеется достаточно развитый математический аппарат теории конечных автоматов. Оценим сложность анализа схем на риски сбоя. Если на вход комбинационной схемы подаётся n переменных, то на нём могут действовать 2^n наборов, от каждого из которых может осуществляться переход к $2^n(2^n-1)$ переходов. При $n \geq 4$ число переходов приблизительно 2^{2n}. Оценим время анализа при следующих числовых данных: количество переменных n=64; ЭВМ способна проанализировать переход между двумя наборами за 1 мкс. Время анализа в данном случае будет составлять $2^{128} \cdot 10^{-6}$ с или приблизительно 10^{25} лет. Этот пример показывает, что рассматриваемый анализ является сложной многоразмерной задачей. Она облегчается в какой-то степени тем, что не всегда $n \geq 64$, а число переходов между различными наборами не превышает, как правило, несколько сотен [11].

Широкое распространение получили следующие методы:

- использование временных диаграмм, в том числе асинхронное моделирование на их основе [7, 8];
- графический метод Хаффмена [9];
- использование многозначной логики [10];
- использование двоичной алгебры;

– методы, основанные на аппарате дифференциальных булевых уравнений.

Временные диаграммы являются эффективным средством анализа переходных процессов в цифровых системах. Построение и анализ временных диаграмм выполняется на ЭВМ с помощью специальных моделирующих программ. Временные диаграммы являются основой при выполнении асинхронного моделирования, однако этот метод требует представления схемы по многоярусной структуре, поэтому риски сбоя выявляются не всегда.

Графический метод Хаффмена разработан для анализа схем с небольшим числом переменных. Анализ проводится по картам Карно и графам переходов наборов. Однако с ростом числа переменных, от которых зависит функция алгебры логики, этот метод становится практически неприемлемым из-за графической громоздкости.

Все методы многозначной логики основаны на использовании, кроме значений 0 и 1 булевой алгебры, различных представлений событийных сигналов:

– при трёхзначном моделировании для представления значений величин сигналов берётся множество L={0, 1/2, 1}, где 0 и 1 интерпретируются так же, как и в булевой алгебре, а 1/2 используется для представления событийного (переходного) процесса. Значение 1/2 воспринимается логическим элементом либо как 0, либо как 1, поэтому при моделировании оно обозначается 1/2, причём это обозначение надо рассматривать как единый символ;
– четырёхзначная модель (алгебра Поста): 0, переходы 01 и 10, 1;
– пятизначная модель: 0, 01, 10, 1, X – неопределённое значение;
– восьмизначная модель: 0, 1, переходы 01 и 10, которые обозначаются специальными символами “+” и “-” соответственно, статические риски сбоя S_0 и S_1, динамические риски сбоя D_+ и D_-;
– девятизначная модель: к символам восьмизначной модели добавляется символ ‘неопределенное значение’, под которым понимается случайное значение выхода RS-триггера, когда на его входах совершается переход от запрещенного набора к набору, соответствующему режиму хранения. Этот метод применяется для анализа на риски сбоя схем с памятью или с обратными связями.

Все методы многозначного моделирования достаточно сложны для ручного применения и рассчитаны, в основном, для проведения анализа схем на ЭВМ. Для ручного применения используют методы трёхзначного и восьмизначного моделирования и только для сравнительно простых схем.

Особенностью метода, использующего двоичную алгебру, является возможность определения не только факта содержания рисков сбоя в схеме на заданных входных переходах, но и вычисления количества возможных ложных переходов на выходах схемы.

В методах, основанных на аппарате дифференциальных булевых уравнений, в булевы функции непосредственно вводится дискретная временная функция, а изменение булевых функций во времени оценивается с помощью производной функции по времени. Алгоритм выполнения анализа схем с помощью этого метода достаточно сложен, но позволяет выявлять соотношения задержек в состязающихся цепях, которые определяют наличие или отсутствие сбоя, то есть возможно получение рекомендаций для корректировки влияния состязаний.

Таким образом, независимо от методов синтеза цифровых схем, разработчики имеют достаточно разнообразных методов анализа цифровых структур для определения их функциональной устойчивости. Проверка результатов моделирования производится с помощью редактора сигналов текстового редактора.

Не менее важное значение при разработке цифровых устройств имеет временной анализ. Временной анализ, как правило, проводится отдельной процедурой - временным анализатором (Timing Analyzer). Временной анализатор должен обеспечивать следующие функции: анализ задержек прохождения сигналов через кристалл (от входа до выхода); анализ регистровой логики проекта: расчет задержек сигналов, минимальной величины периода синхронизации (Clock), максимальной рабочей

частоты кристалла; вычисление минимальных времен установки/сброса, требуемых для прохождения сигнала от контактов кристалла до входов регистров и защёлок.

Существенным аспектом при разработке и моделировании цифрового устройства является выбор соответствующего уровня абстракции. В настоящее время одна БИС может содержать сотни тысяч логических вентилей, поэтому важно правильно выбрать не только способ описания, но и уровень представления системы, построенной на базе подобных БИС. В противном случае разобраться в работе системы будет очень трудно. Ключом к решению этой проблемы является работа на соответствующем уровне абстракции – уровне представления, который дает всю информацию, необходимую в данный момент времени, но избавляет разработчика от необязательных деталей.

Абстрактное представление может относиться к одной из двух областей, которые определяются следующим образом.

Структурная область – область, предусматривающая описание компонента как совокупности взаимосвязанных компонентов более примитивного уровня.

Поведенческая область – область, предусматривающая описание компонента по зависимостям вход/выход при помощи некоторой процедуры. Типичные уровни абстракции для цифровых систем показаны на рис.37[12]. Это уровни процессоров-памяти-коммутаторов (ППК), микросхемный (ИС), регистровый, вентильный, схемный и кремниевый. Иерархия абстракций имеет форму усеченной пирамиды. Расширение пирамиды книзу отображает увеличение степени детализации, т.е. количества деталей, которое должно обрабатываться при представлении СБИС-системы на этом уровне.

В табл.3 приведена характеристика иерархии – указываются примитивы структуры и поведенческое представление для каждого уровня.

Рис.37. Уровни абстракции ЦУ

На самом нижнем уровне – кремниевом – в качестве базовых примитивов используются геометрические формы, которые представляют области диффузии, поликремния и металлизации на поверхности кремниевого кристалла. Здесь представление только структурное - не поведенческое.

На следующем, более высоком уровне – схемном – представление проекта формируется с использованием межсоединений традиционных пассивных и активных элементов электрической схемы: резисторов, конденсаторов, биполярных и МОП-транзисторов. Соединение этих компонентов используется для моделирования поведения электрической схемы, выражаемого через значения

напряжений и токов. Для поведенческого описания на этом уровне можно использовать дифференциальные уравнения.

Третий уровень – уровень логических вентилей традиционно играет основную роль при проектировании ЦУ.

Здесь используются такие базовые примитивы, как логические вентили И, ИЛИ и НЕ и различные типы триггеров. Соединение этих примитивов позволяет образовывать комбинационные и последовательностные логические схемы. Поведенческие описания на этом уровне – булевы формулы.

Выше вентильного уровня в иерархии находится регистровый уровень: здесь базовые примитивы – это такие компоненты, как регистры, счетчики, мультиплексоры и арифметико-логические устройства (АЛУ). При работе на этом уровне используются таблицы истинности и таблицы состояний. Поведенческое представление примитивов регистрового уровня возможно также с использованием языков регистровых передач. Над регистровым уровнем находится уровень микросхем (ИС).

На микросхемном уровне в качестве структурных компонентов выступают микропроцессоры, устройства основной памяти, последовательные порты, параллельные порты и контроллеры прерываний. Хотя границы микросхем обычно являются и границами примитивов моделей, возможны и другие ситуации. Так, набор микросхем, которые совместно образуют одно устройство, можно представить как один примитив (например разрядно-модульный процессор).

Таблица 3

Иерархия моделей ЦУ

Уровень	Структурные примитивы	Поведенческое представление
ППК	Центральные процессоры, памяти, шины	Технические данные и характеристики
Микросхемный (ИС)	Микропроцессоры, ЗУПВ, ПЗУ, УАПП, параллельные порты	Входные-выходные зависимости, алгоритмы, микрооперации
Регистровый	Регистры, АЛУ, счетчики, мультиплексоры	Таблицы истинности, таблицы состояний, микрооперации
Вентильный	Логические вентили, триггеры	Булевы (логические) формулы
Схемный	Транзисторы, резисторы, индуктивности и емкости	Дифференциальные уравнения
Кремниевый	Геометрические объекты	Нет

Возможен и альтернативный вариант, когда примитивы представляют отдельные секции одной микросхемы, например на этапе анализа технического задания и декомпозиции. Главной особенностью здесь является то, что примитивом представляется большой блок логики, где для длинных и зачастую сходящихся трактов обработки данных необходимо представлять зависимости выходов от входов.

Как и в случае примитивов нижележащих уровней, примитивы микросхемного уровня не строятся иерархически из более простых примитивов, а представляют собой единые объекты-модели. Поведенческое описание модели микросхемного уровня строится на основе входной-выходной зависимости каждой конкретной ИС – алгоритма, реализуемого данной ИС. Верхний уровень иерархии структур модели – это уровень процессоров – памяти – коммутаторов (ППК). В качестве примитивов этого уровня используются процессор, память и коммутатор (шина). Поведенческое описание на этом уровне включает такие основные данные и характеристики, как, например, показатель быстродействия процессора в миллионах команд в секунду или пропускная способность тракта обработки данных (бит/с).

Из табл.3 и вышеизложенного видно, что структурные и поведенческие характеристики соседних уровней в определенной степени перекрываются. Например, и на регистровом, и на микросхемном уровне может использоваться представление при помощи микроопераций. Однако структурное представление для обоих уровней совершенно различно. Микросхемный уровень и уровень ППК имеют одни и те же примитивы, однако абсолютно различны по своим поведенческим характеристикам. Так, поведенческие модели уровня ИС позволяют вычислять детальные ответные реакции в виде целых чисел и битов. А поведенческому представлению уровня ППК свойственно серьезное ограничение – оно служит преимущественно для моделирования пропускной способности системы или построения стохастической модели. На практике ППК используется главным образом для классификации компьютерных архитектур.

ПОДГОТОВКА К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Моделирование является не менее важным этапом проектирования ПЛИС, чем ввод и компиляция проекта. Успешная компиляция даёт разработчику гарантии лишь того, что созданный файл прошивки кристалла соответствует входному описанию. Моделирование является наиболее простым (с точки зрения временных и материальных затрат) способом проверки проекта на правильность функционирования.

Перед моделированием разработчик создаёт файл входных воздействий (Simulator Channel File). Симулятор использует эти воздействия для создания формы выходных сигналов в том виде, в каком они были бы получены на реальном кристалле при тех же условиях.

В зависимости от информации, необходимой разработчику, может быть выполнено функциональное, временное или ‘связанное’(для нескольких проектов) моделирование. Функциональное моделирование используется для проверки правильности функционирования устройства на ранних стадиях разработки (до синтеза кристалла). Временное моделирование проводят на заключительной стадии разработки с учётом временных характеристик реального устройства.

Наибольший интерес для разработчика представляет временное моделирование, поскольку оно позволяет определить быстродействие, состязания сигналов, риски сбоя разработанного устройства.

Для моделирования проекта нужно выполнить следующую последовательность шагов:

1) создать файл входных воздействий;
2) добавить в файл необходимые входные и выходные сигналы;
3) задать форму входных сигналов;
4) объединить шинные сигналы в группы;
5) сохранить и закрыть файл;
6) открыть окно симулятора;
7) определить дополнительные выходные файлы;
8) включить контроль времён установки/сброса сигналов;
9) запустить симулятор;
10) создать табличный файл;
11) проверить результаты моделирования.

Создание файла входных воздействий

Выбрать команду ‘New’ (File menu). Появляется диалоговое окно, в котором следует выбрать ‘Waveform Editor File’ и расширение – ‘scf’. Нажать ‘Ok’. Появляется окно ‘Waveform Editor’ – редактор сигналов.

Выбрать ‘End Time’ (File menu) и установить время моделирования. По умолчанию установлено ‘1.0 us’ (1 мкс – максимальное время моделирования для системы MAX PLUS II).

Выбрать ‘Grid Size’ (Options menu) и установить шаг временной сетки 50 ns.

Выбрать ‘Enter Nodes from SNF’ (Node menu). Появляется диалоговое окно (рис.38).

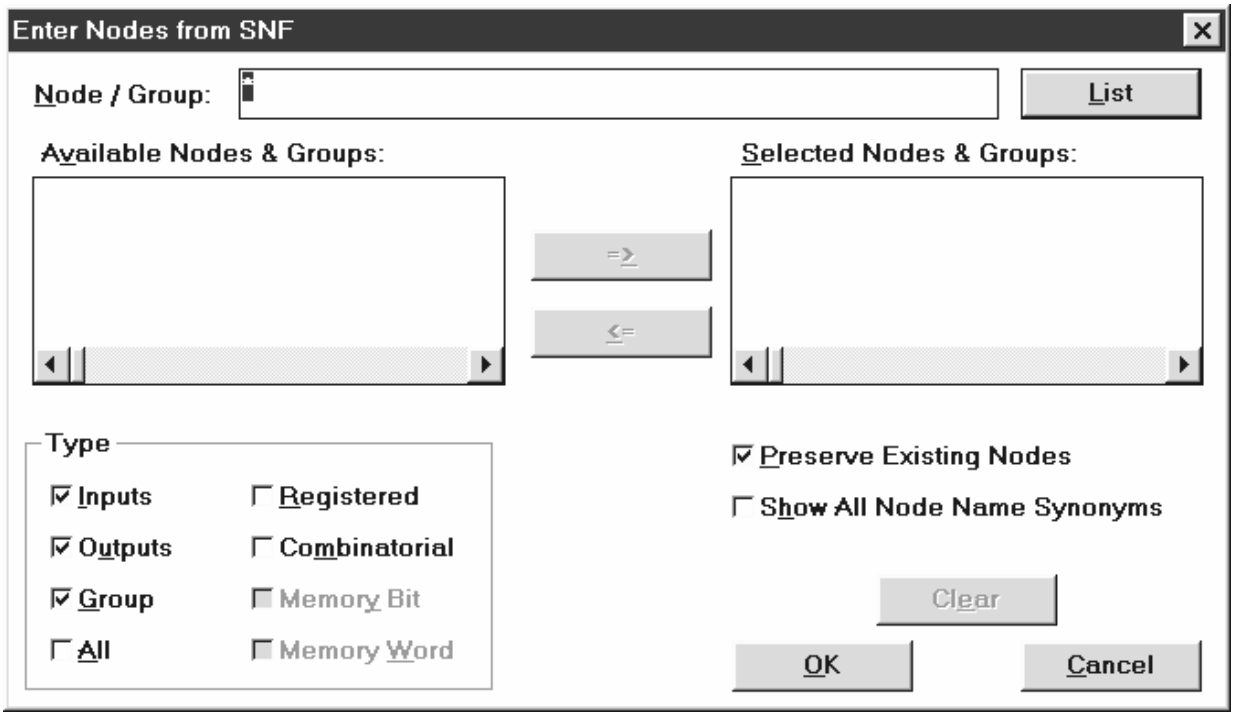


Рис.38. Меню ‘Enter Nodes from SNF’

Выбрать команду ‘List’. В окне ‘Available Nodes & Groups’ все доступные входные/выходные сигналы и группы сигналов (шины).

Левой кнопкой мыши пометить нужные сигналы и с помощью кнопки со стрелкой скопировать их в окно ‘Selected Nodes & Groups’. Нажать ‘Ok’. В окне редактора сигналов появляются выбранные сигналы.

Выбрать команду ‘Save as’ (File menu). Появляется диалоговое окно, в котором автоматически высвечивается текущее имя проекта с расширением .scf. Нажать ‘Ok’.

Добавление сигналов и групп сигналов в файл .scf

Имеется возможность добавлять дополнительные сигналы для моделирования с помощью команды ‘Insert Node’ (Node menu) (рис.39).

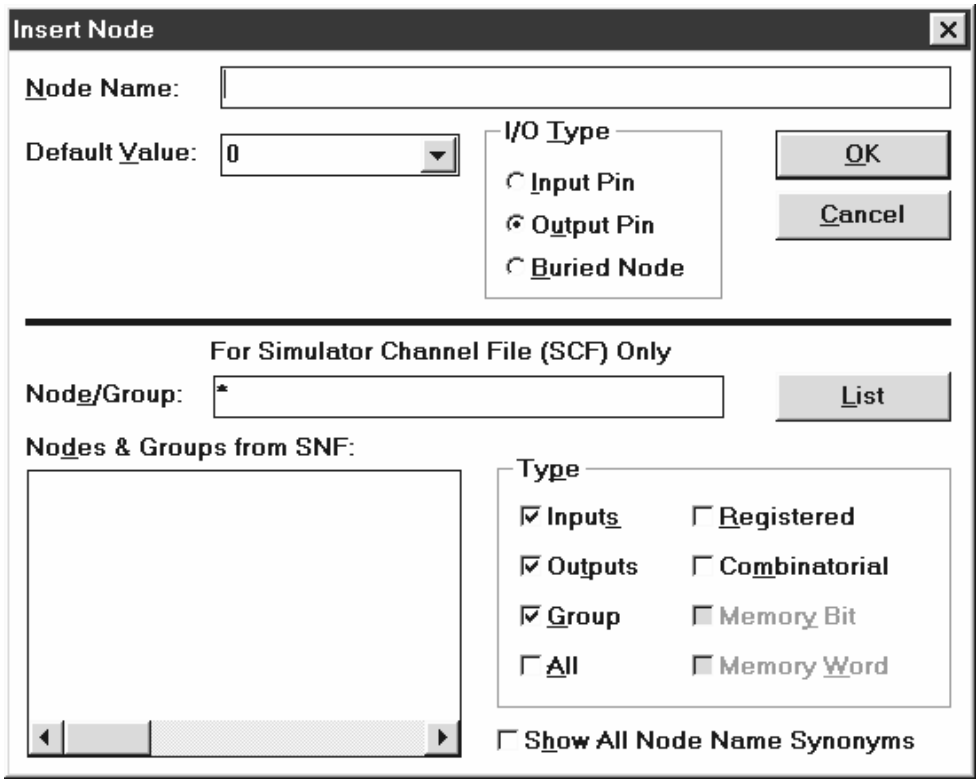


Рис.39. Меню ‘Insert Node’

Редактирование формы входных сигналов

Разработчик может отредактировать форму входных сигналов, чтобы задать входные воздействия для моделирования. При моделировании симулятор автоматически создаёт форму выходных сигналов в соответствии с входными. Редактирование уровня сигналов осуществляется с помощью кнопок ‘Edit menu’, расположенного в левой части окна:

- установка в ‘0’;
- установка в ‘1’;
- установка в неопределённое состояние (X);
- установка в третье состояние (Z);
- инвертирование сигнала;
- установка формы тактовых сигналов (Clock).

Перед редактированием сигнал (или его часть) нужно пометить с помощью мыши. Общий вид редактора сигналов показан на рис.40.

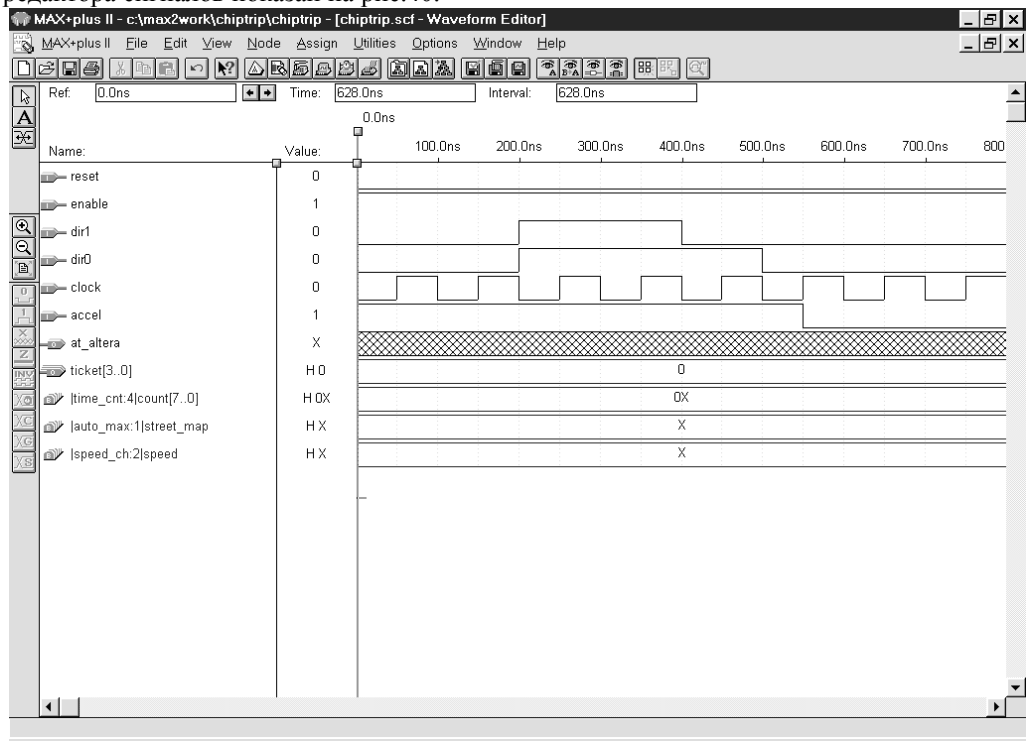


Рис.40. Редактор сигналов системы MAX PLUS II

Запуск симулятора

Симулятор использует в качестве входных данных файл .scf.

- Выбрать команду ‘Simulator’ (MAX+PLUS II menu). Появляется окно симулятора (рис.41). Список цепей для моделирования (.snf - Simulator Netlist file) и файл входных воздействий (.scf – Simulator Channel file) для текущего проекта загружаются автоматически.

Определение дополнительных выходных файлов

Выбрать команду ‘Inputs/Outputs’ (File menu). Появляется диалоговое окно, в котором нужно включить опции ‘History (.hst)’ и/или ‘Log (.log)’: эти файлы автоматически будут созданы при моделировании.

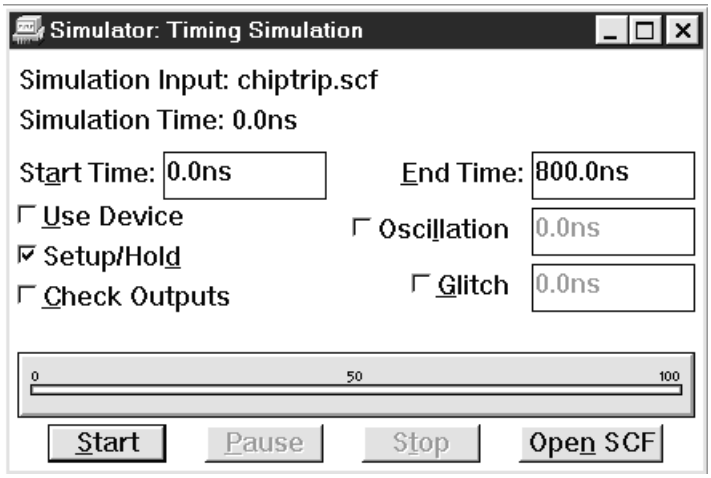


Рис.41. Окно симулятора

Файл .hst содержит все команды и опции, которые использовались при моделировании, а также сообщения симулятора. Файл .log содержит все команды симулятору и используется обычно в качестве командного файла для повторного моделирования.

Нажать ‘Ok’.

Контроль времён установки/сброса сигналов

Включение опции ‘Setup/Hold’ в окне симулятора позволяет контролировать нарушения времени установки/сброса сигналов.

Запуск симулятора

Нажать кнопку ‘Start’ в окне симулятора. Моделирование начинается немедленно. При завершении моделирования выдаётся сообщение о количестве ошибок, сообщений, времени моделирования и стабилизации схемы.

Создание табличного файла

Табличный файл представляет собой текстовую альтернативу графическому .scf – файлу, в котором представлены результаты моделирования.

Выбрать ‘Create Table File’ (File menu). Появляется диалоговое окно.

Выбрать тип расширения - .tbl и нажать ‘Ok’.

Проверка результатов моделирования

Проверка результатов моделирования производится с помощью редактора сигналов (рис.42) и текстового редактора.

Выбрать ‘Open SCF’ в окне симулятора, при этом открывается редактор сигналов с текущим файлом .scf.

С помощью команд ‘Zoom In’, ‘Zoom Out’ и прокрутки экрана проверить результаты моделирования.

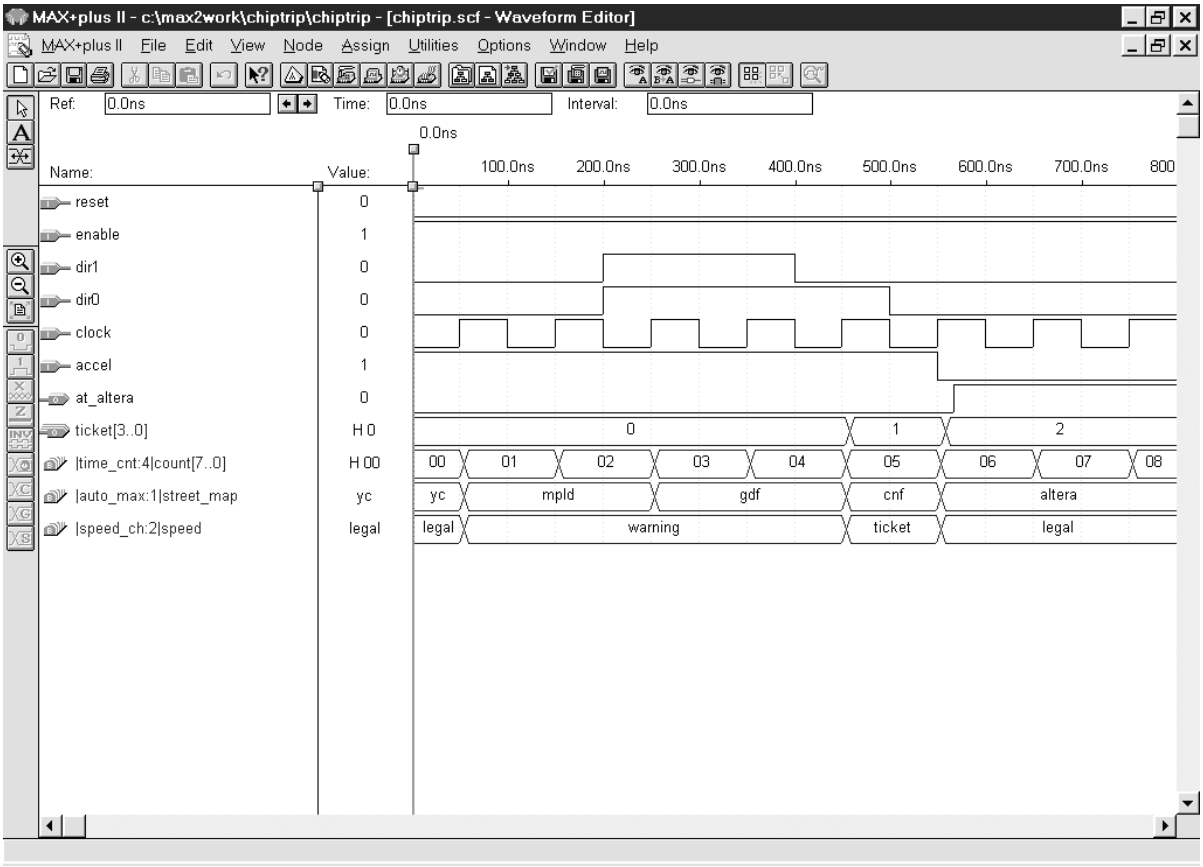


Рис.42. Результаты моделирования в редакторе сигналов

Для изучения текстовых файлов отчёта о результатах моделирования необходимо выбрать ‘Open’ (File menu), в появившемся диалоговом окне выбрать ‘Text Editor files’ и тип расширения (.hst, .log, .tbl).

Для повторного моделирования следует отредактировать входные сигналы и повторить цикл заново.

Временной анализ

В системе MAX PLUS II временной анализ проводится временным анализатором (Timing Analyzer). Поддерживаются три режима работы:
‘Delay Matrix’, ‘Registered Performance’ и ‘Setup/Hold Matrix’.

‘Delay Matrix’ – анализируются задержки прохождения сигналов через кристалл (от входа до выхода).

‘Registered Performance’ – анализируется регистровая логика проекта: рассчитываются задержки сигналов, минимальная величина периода синхронизации (Clock), максимальная рабочая частота кристалла.

‘Setup/Hold Matrix’ – вычисляются минимальные времена установки/сброса, требуемые для прохождения сигнала от контактов кристалла до входов регистров и защёлок.

- Выбрать ‘Timing Analyzer’ (MAX+PLUS II menu).
- Выбрать режим работы анализатора (Analysis menu).

Если выбран режим ‘Delay Matrix’, то следует обратить внимание на опции ‘Cut Off I/O Pin Feedback’ и ‘Cut Off Clear & Preset Paths’ (Options menu).

Если опция ‘Cut Off I/O Pin Feedback’ включена, то двунаправленные выводы кристалла анализируются как входы и выходы: обратные связи отсутствуют.

Если опция ‘Cut Off Clear & Preset Paths’ выключена, анализатор вычисляет время прохождения сигналов установки/сброса до входов D – триггеров.

- Нажать кнопку ‘Start’.

На рис.43 показан временной анализатор в режиме ‘Delay Matrix’.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Составить осциллограммы работы схемы.
2. Создать на основе составленных осциллограмм файл входных воздействий.
3. Произвести моделирование проекта для контроля всех режимов работы.
4. Произвести временной анализ проекта во всех режимах работы временного анализатора.

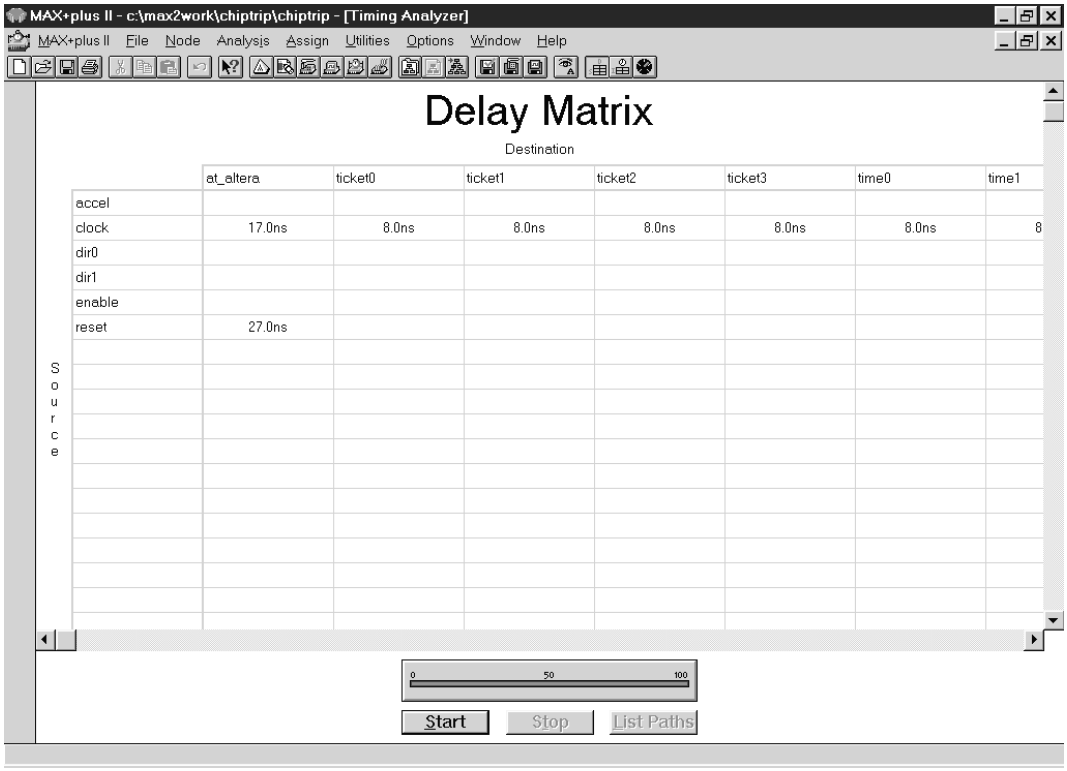


Рис.43. Временной анализатор в режиме ‘Delay Matrix’

СОДЕРЖАНИЕ ОТЧЁТА

1. Краткие теоретические сведения о моделировании и временном анализе цифровых устройств.
2. Протокол работы с симулятором.
3. Протокол работы с временным анализатором.
4. Анализ полученных результатов.
5. Выводы о характеристиках разработанного устройства.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего проводится моделирование цифровых устройств? Основные характеристики алгоритмов моделирования.
2. Основные методы моделирования: достоинства и недостатки.
3. Проанализировать триггерную схему методом трехзначного моделирования.
4. Какие задачи решаются на этапе временного анализа проекта? При каких условиях может быть вычислена максимальная рабочая частота устройства?
5. Какое влияние на разработку и моделирование оказывает выбор уровня абстракции представления проекта?
6. Каковы недостатки модуля моделирования системы MAX PLUS II?

ПРИМЕР

После компиляции проекта разработчику необходимо убедиться в правильности функционирования устройства и проверить его рабочие характеристики. Прежде всего нужно создать

файл входных воздействий (*.scf), в котором задать форму входных сигналов. Как правило, при большом количестве входных сигналов задаются не все сигналы, а только те, которые позволят проконтролировать выполнение определенной функции разработанного устройства.

1) Запускаем ‘Waveform Editor’ из меню ‘Max+Plus II’.
2) В меню ‘Node’ выбираем команду ‘Insert Node’.
3) Задаем имя входного сигнала – ‘In’ и определяем его тип – ‘Input’.
4) Аналогично задаем имена выходных сигналов (значения которых нужно проконтролировать), в нашем случае – Out0, Out1, Out2, Out3.

В результате получаем окно с заданными сигналами (рис.44). Далее необходимо задать форму сигнала In.

5) Выделяем сигнал In левой кнопкой мыши. Становится активным меню форм сигналов в левой части окна. С помощью кнопки задаем периодичность сигнала In: во вкладке ‘Increment By’ появившегося меню ‘Overwrite Count Value’ указываем ‘2’. Поскольку шаг временной сетки по умолчанию равен 50 ns, период сигнала In становится равным 100 ns (рис.45).

6) Запоминаем файл – по умолчанию ему дается имя проекта.

7) Моделируем проект. Запускаем ‘Simulator’ (меню ‘MAX+Plus II’).

8) После моделирования получаем отклик (форму) выходных сигналов на заданные входные воздействия (рис.46).

9) Если полученная форма сигналов соответствует заданным, можно провести временной анализ проекта.

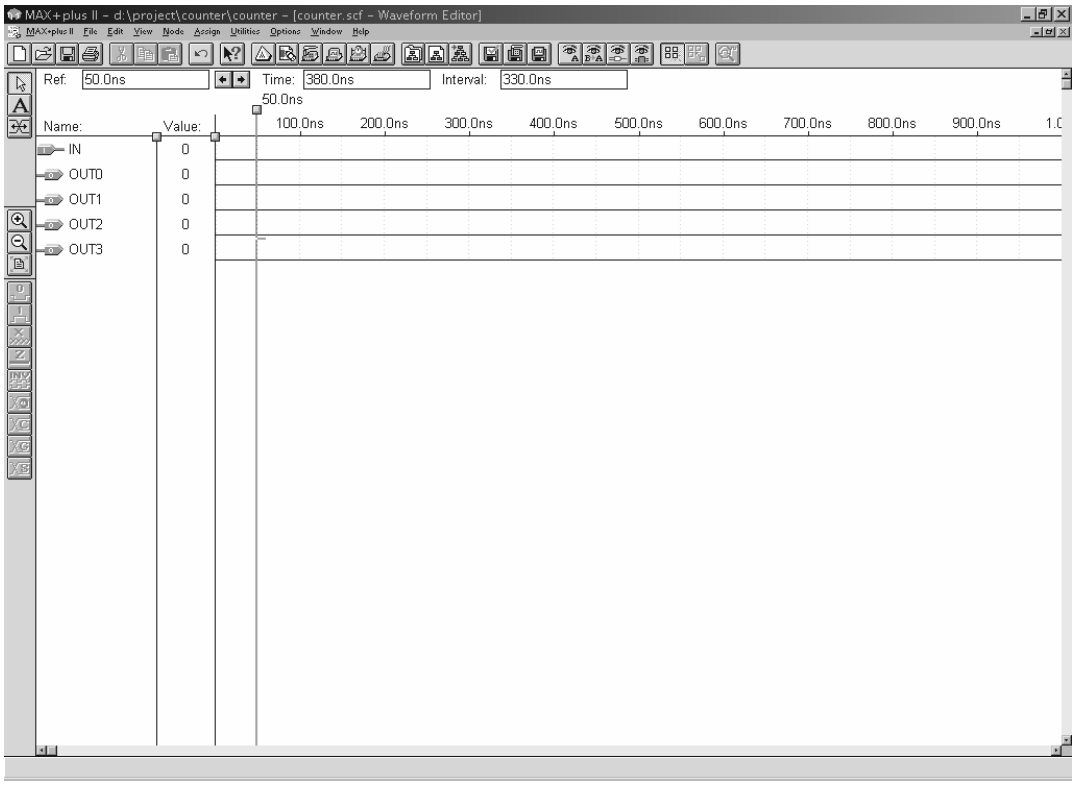


Рис.44. Редактор сигналов до задания формы сигнала In

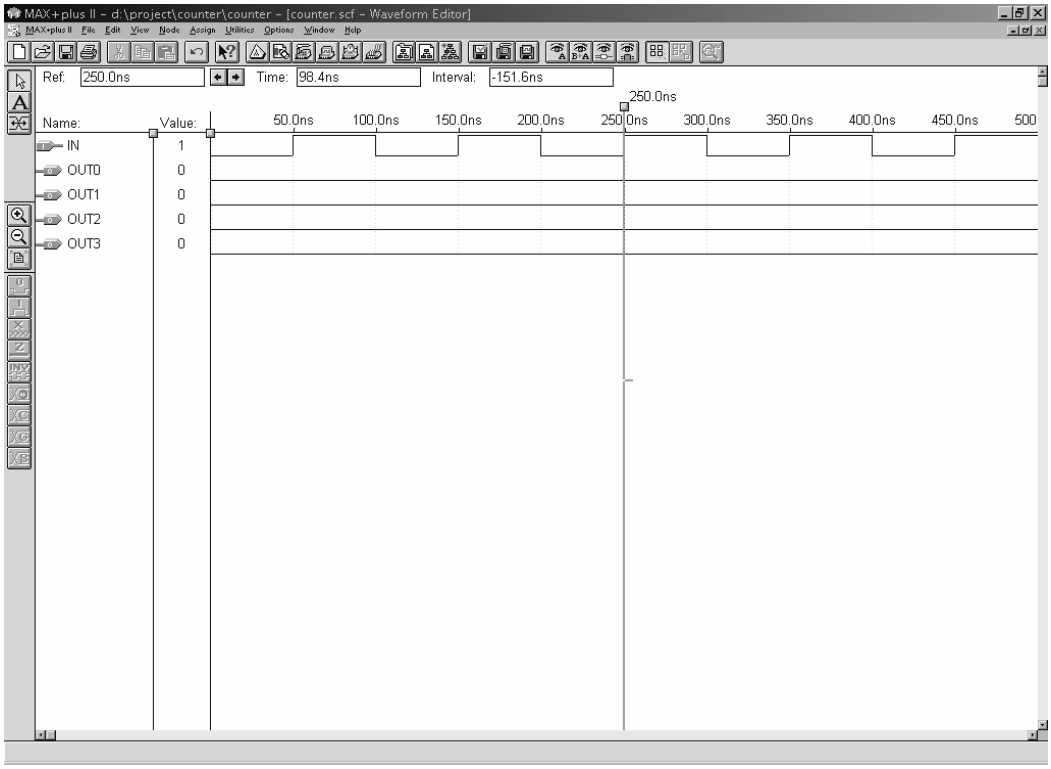


Рис.45. Редактор сигналов после задания формы сигнала In

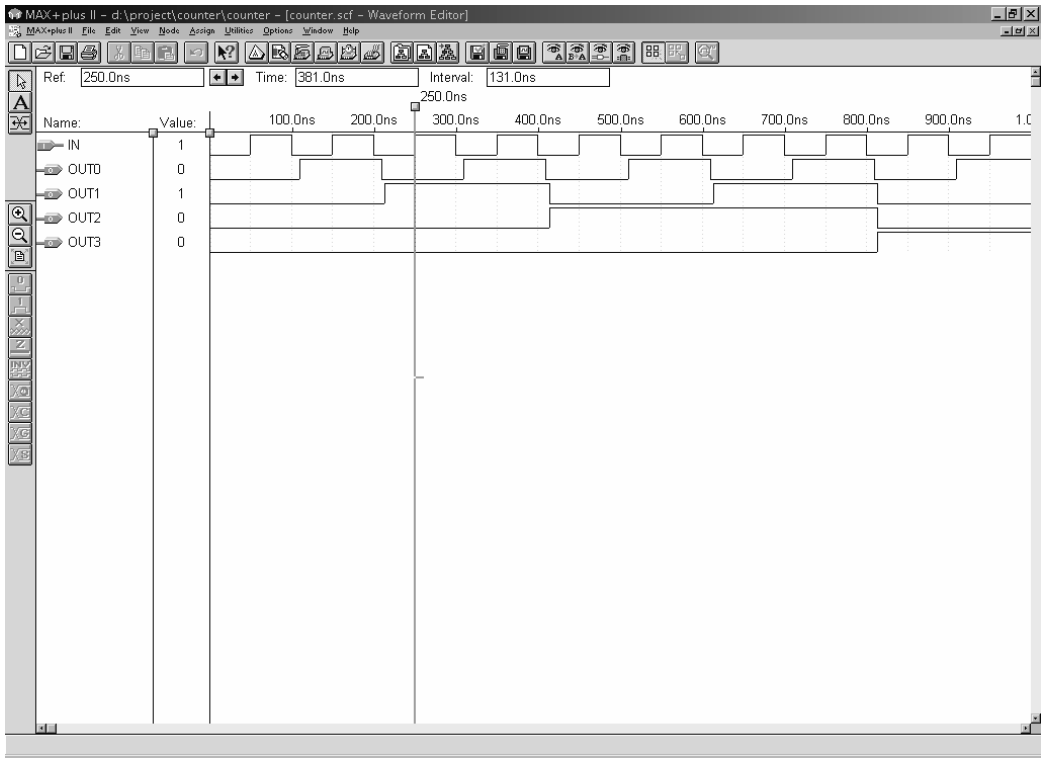
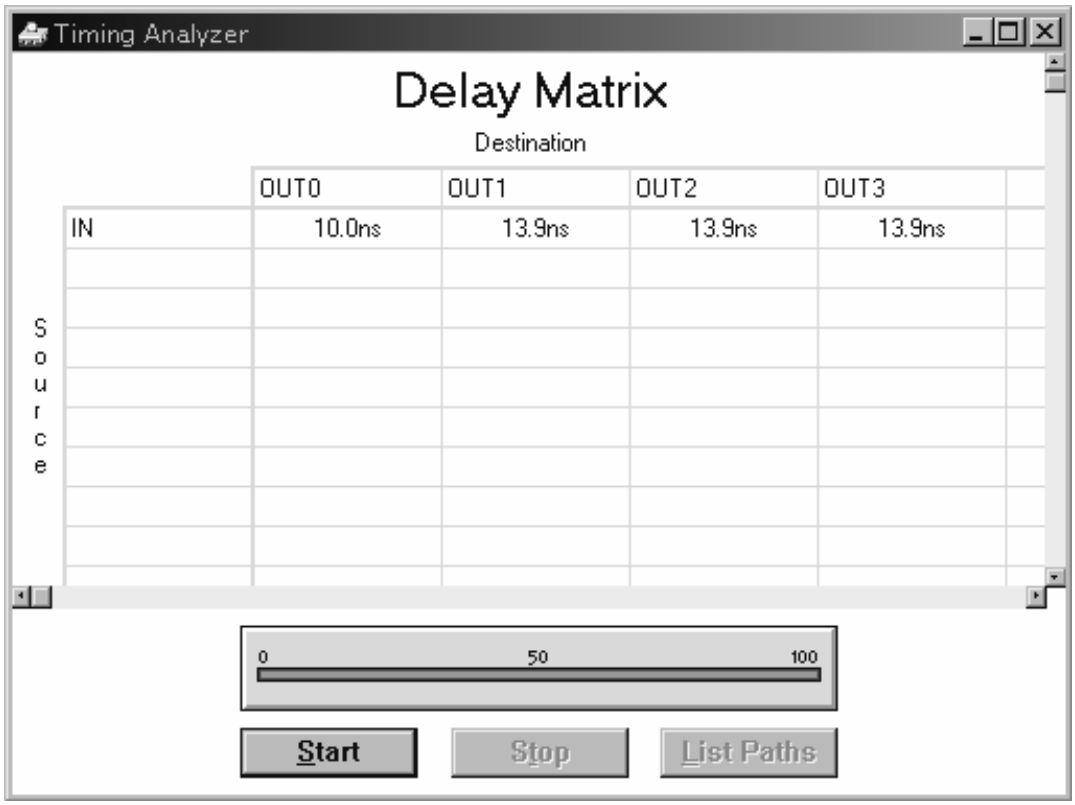


Рис.46. Редактор сигналов после моделирования

Проведем временной анализ в двух режимах работы: ‘Delay Matrix’ и ‘Registered Performance’. На рис.47 показаны задержки прохождения сигналов на кристалле для проекта Counter. На рис.48 показаны характеристики быстродействия разрабатываемого устройства: максимальная рабочая частота – 172,41 MHz, минимальная величина периода синхронизации – 5,8 ns.



Результаты временного анализа в режиме Delay Matrix. Рис.47

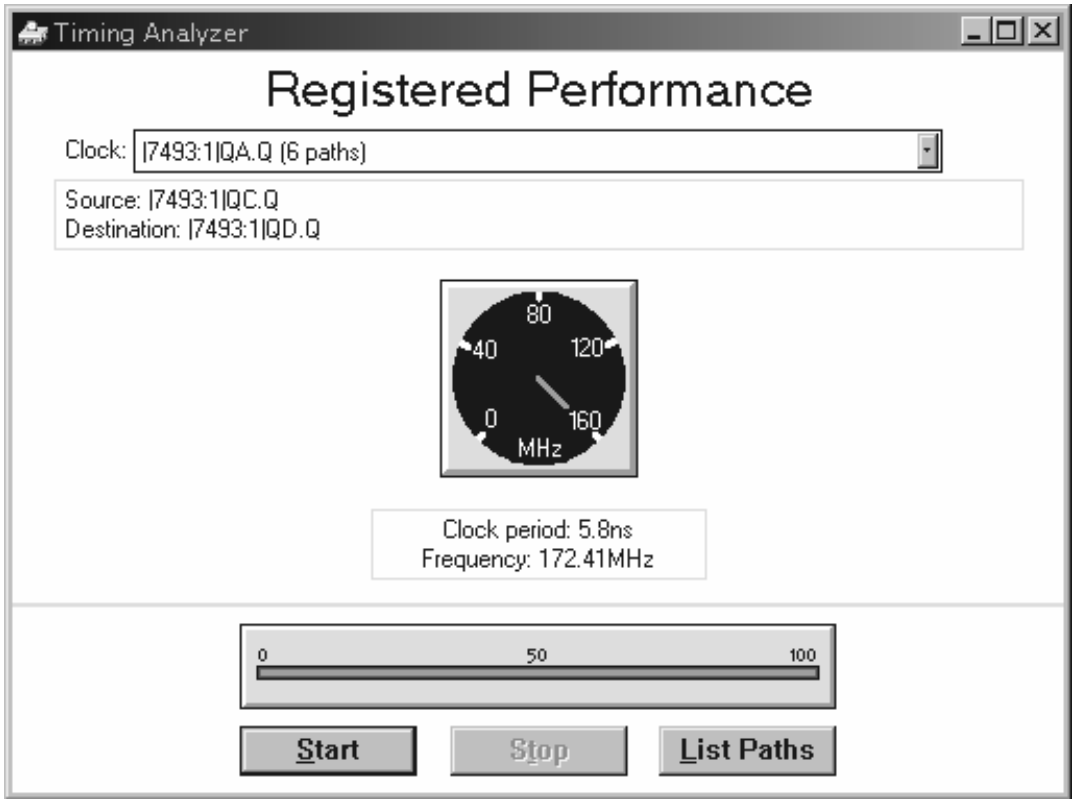


Рис.48. Результаты временного анализа в режиме Registered Performance

4.5. Лабораторная работа №5. Программирование ПЛИС

Цель работы: изучение методики программирования ПЛИС, получение навыков работы с ByteBlaster -ом.

ВВЕДЕНИЕ

В процессе проектирования ПЛИС САПР формирует файлы конфигурации. В течение длительного времени единственным форматом таких файлов являлся международный стандарт JEDEC, обеспечивающий совместимость с устройствами прошивки ПЛИС – программаторами. Однако для некоторых новых микросхем используются либо специфические (как для ПЛИС семейств MAX7000/9000), либо стандартные (например, шестнадцатеричные HEX для ПЛИС семейств FLEX 8000/10K) форматы.

Завершающим этапом изготовления специализированной БИС на основе ПЛИС является загрузка конфигурации в микросхему. В зависимости от технологии ПЛИС различают два варианта реализации этого процесса: технологическое программирование и инициализация.

Технологическое программирование производится для ПЛИС, изготовленных по технологии ТТЛШ или КМОП с ультрафиолетовым (EPROM) или электрическим (EEPROM) стиранием и обладающих способностью сохранять конфигурацию (прошивку) при отсутствии напряжения питания. Как правило, технологическое программирование требует наличия программатора, что увеличивает стоимость применения ПЛИС и снижает возможности перехода на новые типы микросхем. Для решения этой проблемы ведущие производители ПЛИС начали выпуск микросхем по технологии EEPROM или FLASH, программируемых непосредственно на печатной плате (ISP – in system programmable) благодаря встроенному порту в стандарте JTAG (IEEE Std. 1149.1 Joint Test Action Group). Программирование и

стирание таких ПЛИС осуществляется через четырёхразрядный интерфейс, обычно подключаемый к последовательному или параллельному порту компьютера (табл.4).

Функцией ISP обладают ПЛИС семейств MAX9000, MAX7000S, FLASHlogic. Следует отметить, что ПЛИС с энергонезависимой прошивкой выдерживают в среднем 100 циклов стирания/записи.

Инициализация характерна для ПЛИС, изготовленных по технологии статического ОЗУ (SRAM): они сохраняют конфигурацию только при наличии напряжения питания и выдерживают неограниченное число циклов перепрограммирования (более 100000). Поэтому при применении таких устройств необходимо предусмотреть возможность загрузки конфигурации в ПЛИС после включения устройства.

Как правило, существуют два протокола этой процедуры: активный и пассивный. В активном режиме данные хранятся в последовательном или параллельном ПЗУ или флэш-памяти, а процессом инициализации управляет сама ПЛИС;

Таблица 4

Разряд	*Описание*	*Функция*
TDI	*Вход данных*	Последовательный вход данных; данные сдвигаются по переднему фронту TCK.
TDO	*Выход данных*	Последовательный выход данных; данные сдвигаются по заднему фронту TCK.
TMS	*Выбор режима*	Вход контроля конечного автомата, реализующего стандарт IEEE 1149.1 JTAG. Данные обновляются по переднему фронту TCK.
TCK	*Тактовый сигнал*	Обеспечивает синхронизацию JTAG-интерфейса; максимальная тактовая частота 10 MHz. Вход находится в состоянии логической единицы в процессе нормального функционирования микросхемы.

в пассивном – функции загрузки обеспечивает внешнее интеллектуальное устройство (процессор или микроконтроллер), частным случаем которого может быть и сам компьютер. Инициализация свойственна ПЛИС семейств FLEX8000, FLEX10K. Микросхемы фирмы Altera поддерживают шесть различных схем конфигурации памяти (табл.5).

Таблица 5

Режим конфигурации	*Источник данных*
Активная последовательная	Последовательное ПЗУ
Активная параллельная с ростом адресов	Параллельное ПЗУ (EPROM, EEPROM, FLASH)
Активная параллельная с уменьшением адресов	
Пассивная последовательная	Последовательное ПЗУ; специализированные аппаратные средства*; контроллер системы, в которую интегрирована ПЛИС
Пассивная параллельная синхронная	Контроллер системы, в которую интегрирована ПЛИС
Пассивная параллельная асинхронная	

* - к специализированным аппаратным средствам относятся загрузочные кабели фирмы Altera: BitBlaster, ByteBlaster, а также FLEX DownLoad Cable, подключаемый к фирменному программатору ASAP.

Наиболее простым и доступным разработчикам средством прошивки ПЛИС является загрузочный кабель ByteBlaster (рис.49).

ByteBlaster поддерживает два режима работы:

- пассивный последовательный (passive serial – PS Mode) - для семейств FLEX10K, FLEX8000);
- JTAG Mode - для семейств FLEX10K, MAX9000, MAX7000S, MAX7000A.

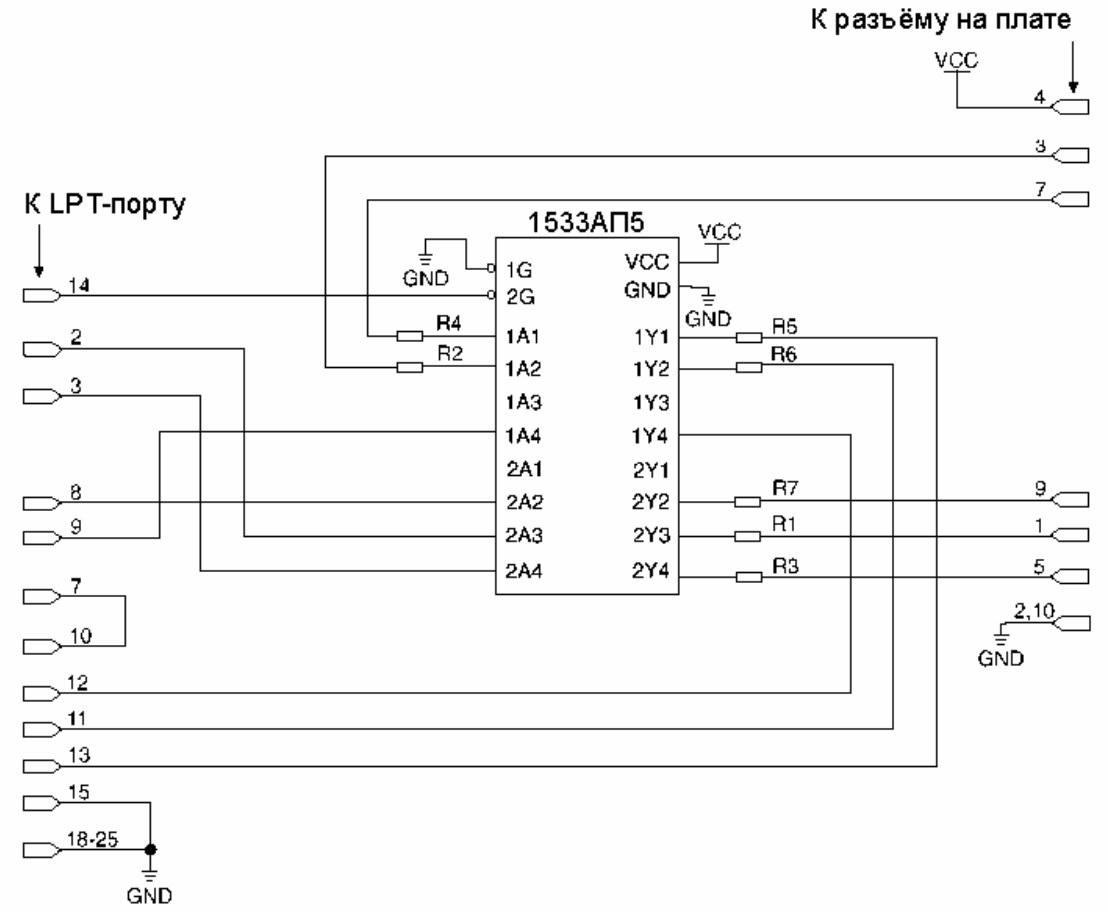


Рис.49. Схема ByteBlaster`a

В табл.6 приведены имена сигналов, которые должны быть подсоединены к 10-pin разъёму, установленному на плате при разных режимах ByteBlaster`a.

Одиночные кристаллы FLEX10K, FLEX8000 могут быть сконфигурированы в режиме PS Mode по схеме на рис.50.

В микросхемы загружается SRAM Object File (.sof), создаваемый автоматически при компиляции проекта. Если вывод DATA0 используется в проекте, то он должен быть изолирован во время конфигурации кристалла.

Для программирования микросхем FLEX10K в режиме JTAG Mode на печатной плате должны быть обеспечены соединения по схеме рис.51.

Таблица 6

Контакт	JTAG Mode	PS Mode
2	TCK	DCLK
3	TMS	Nconfig
8	TDI	DATA0
11	TDO	CONF_DOWN
13	NC	Nstatus
15	GND	GND
18-25	GND	GND

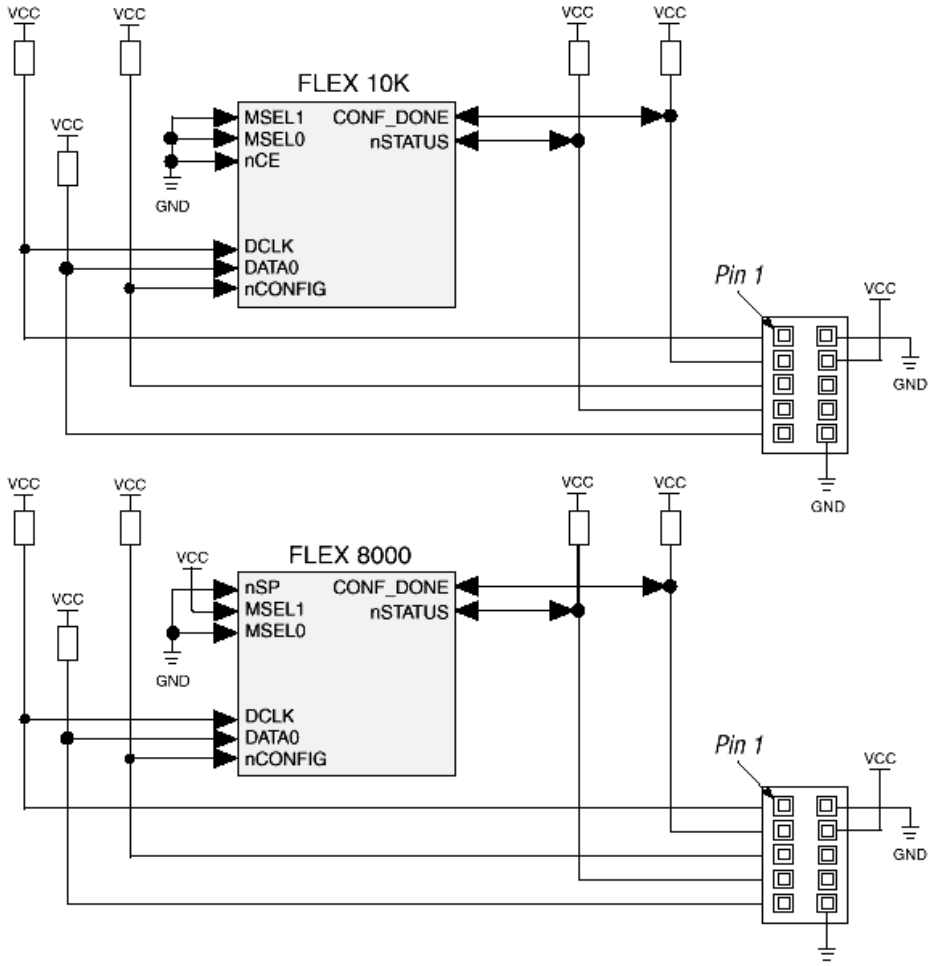


Рис.50. Режим PS Mode

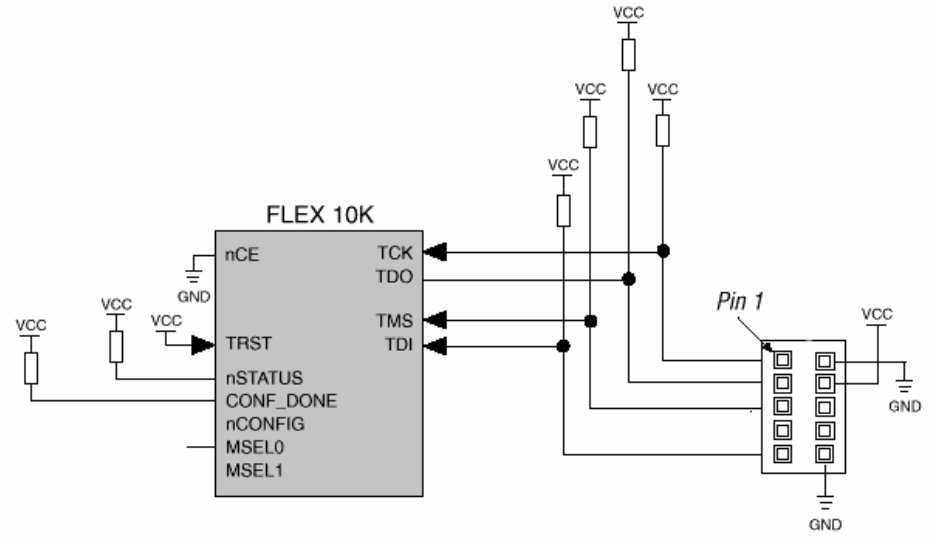


Рис.51. Режим JTAG Mode

С помощью ByteBlaster`а могут быть запрограммированы также все семейства ПЛИС с энергонезависимой прошивкой, поддерживающие интерфейс JTAG: MAX9000, MAX7000S, MAX7000A (рис.52). В микросхемы загружается Programmer Object File (.pof), создаваемый автоматически при компиляции проекта. В процессе программирования все входные/выходные контакты переводятся в третье состояние.

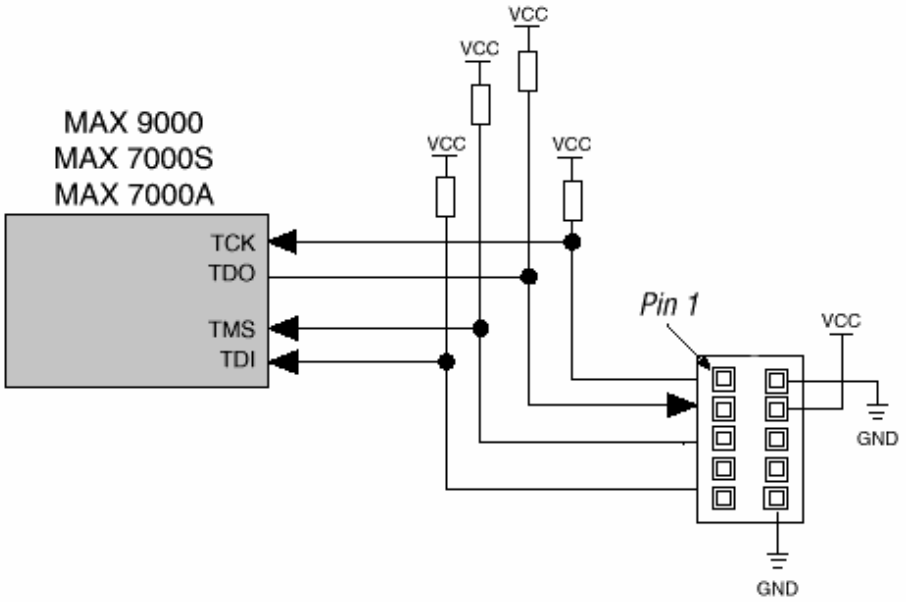


Рис.52. Программирование микросхем семейств MAX7000S/7000A/9000

ПОДГОТОВКА К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Для программирования ПЛИС с помощью программатора (модуль Programmer) системы MAX PLUS II необходимо выполнить следующую последовательность шагов:

- скомпилировать проект;
- подсоединить ByteBlaster;
- открыть окно программатора;

- создать выходной файл работы программатора;
- запрограммировать кристалл.

Компиляция проекта

Компилятор системы MAX PLUS II автоматически создаёт файлы конфигурации *.sof - для кристаллов семейств FLEX10K, FLEX8000, FLEX6000 и *.pof - для MAX9000, MAX7000S, MAX7000A.

Подключение ByteBlaster`a

ByteBlaster подключается к параллельному порту (Lpt) компьютера и к 10-контактному разъёму, установленному на печатной плате. Печатная плата должна обеспечить питание ByteBlaster`a. Следует помнить, что при подключении ByteBlaster`a компьютер должен быть выключен.

Программатор системы MAX PLUS II

Для запуска программатора необходимо выбрать команду ‘Programmer’ (MAX+PLUS II menu). Открывается окно программатора (рис.53). При этом автоматически загружается файл прошивки текущего проекта (вкладка File). Если по каким-то причинам имя файла отсутствует, его можно выбрать с помощью команды ‘Select Programming File’ (File menu).

Рис.53. Окно программатора

Создание выходного файла работы программатора

MAX PLUS II записывает все действия и сообщения программатора в Programmer Log File (.plf). Для создания выходного plf-файла нужно выбрать команду ‘Inputs/Outputs’ (File menu) и в появившемся окне выбрать опцию ’Log(.plf)’.

Программирование кристалла

Программирование кристалла осуществляется с помощью кнопки Program. При нажатии кнопки Verify происходит проверка исходного файла прошивки и информации, содержащейся в ПЛИС.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Произвести компиляцию проекта.
2. Подключить ByteBlaster.
3. Запрограммировать кристалл.
4. Сделать выводы о достоинствах и недостатках ПЛИС.

СОДЕРЖАНИЕ ОТЧЁТА

1. Краткие теоретические сведения о процессе и технических средствах конфигурации ПЛИС.
2. Протокол работы с программатором.
3. Выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сравнительная характеристика режимов конфигурации ПЛИС.
2. Какие преимущества дает разработчику использование загрузочного кабеля ByteBlaster?
3. На каких типах ПЛИС может быть установлен «бит секретности»?
4. Какие функции выполняет программатор системы MAX PLUS II?
5. Сколько циклов записи/стирания выдерживают различные типы ПЛИС?

ПРИМЕР

1. После компиляции проекта выбираем в меню ‘Max+plus II’ опцию ‘Programmer’. Открывается окно программатора.
2. В меню ‘Options’ выбираем команду ‘Select Device’ (рис. 54) и указываем тип микросхемы, если он не был установлен по умолчанию.
3. В меню ‘Options’ выбираем опцию ‘Hardware Setup’ (рис.55) и определяем тип загрузочного устройства. В нашем случае – ByteBlaster.
4. Загружаем файл прошивки в микросхему с помощью кнопки ‘Program’ в окне программатора.

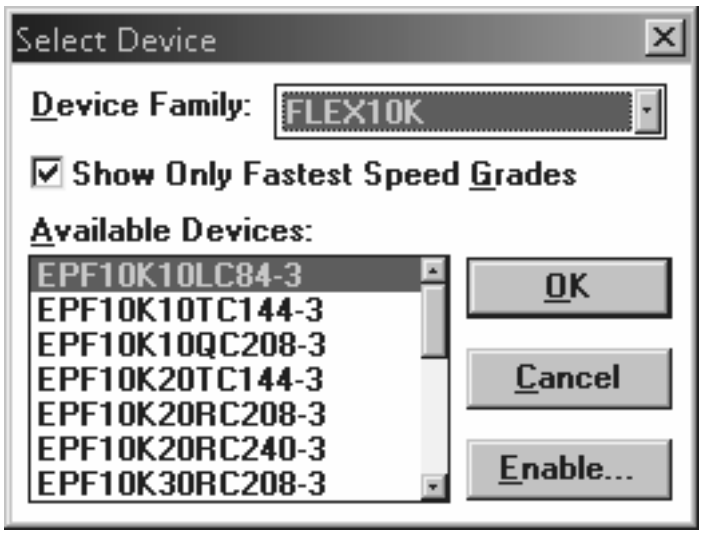


Рис.54. Меню выбора прибора

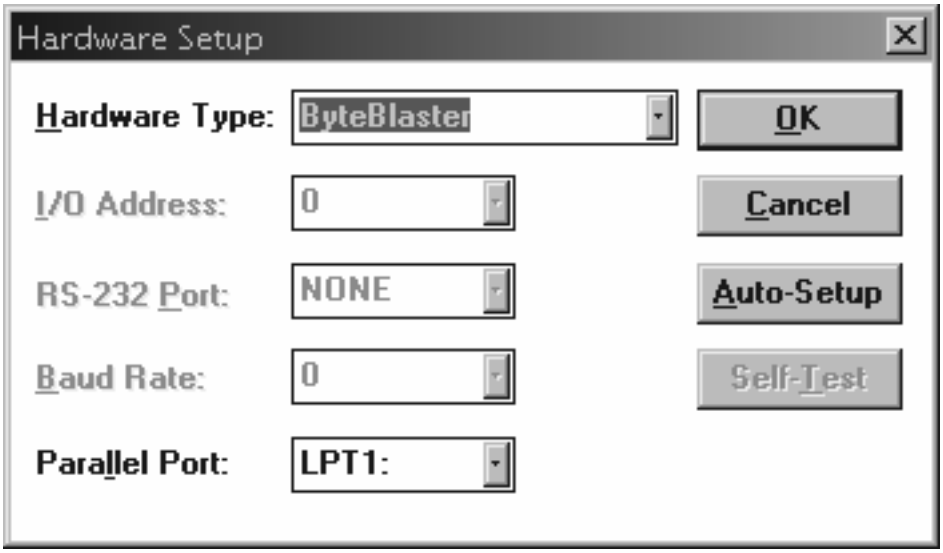
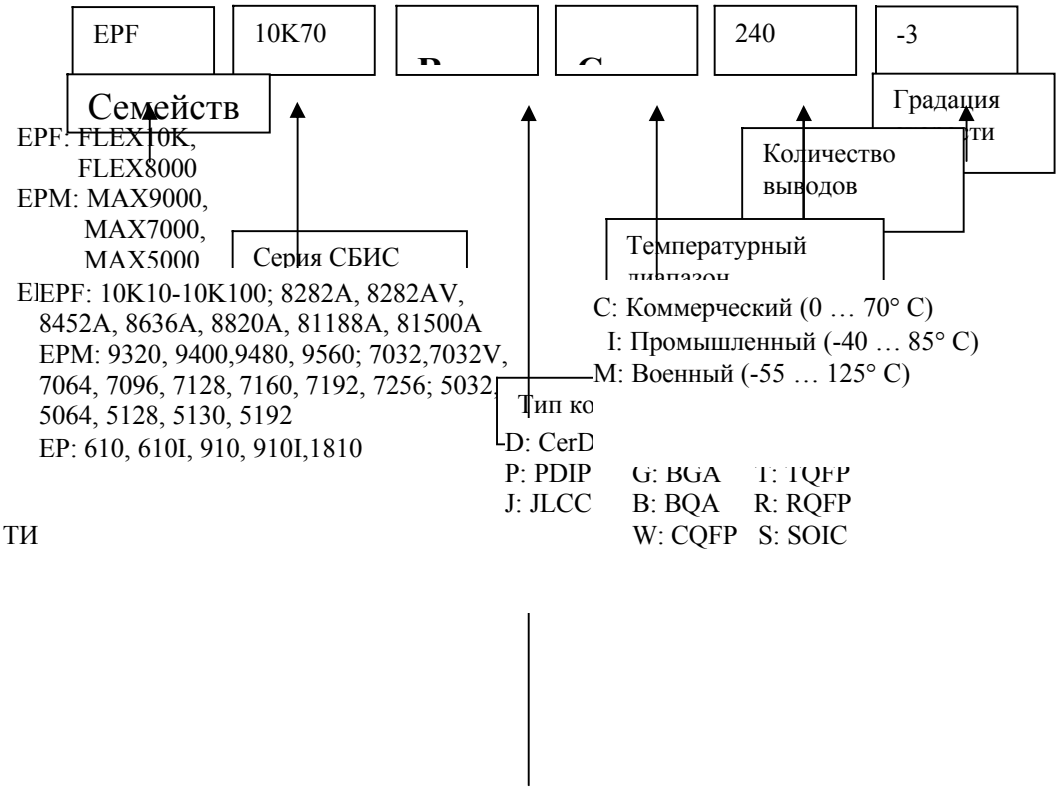


Рис.55. Опция ‘Hardware Setup’

ПРИЛОЖЕНИЕ 1

СТРУКТУРА ОБОЗНАЧЕНИЙ ПЛИС ФИРМЫ ALTERA



ТИ

ПРИЛОЖЕНИЕ 2

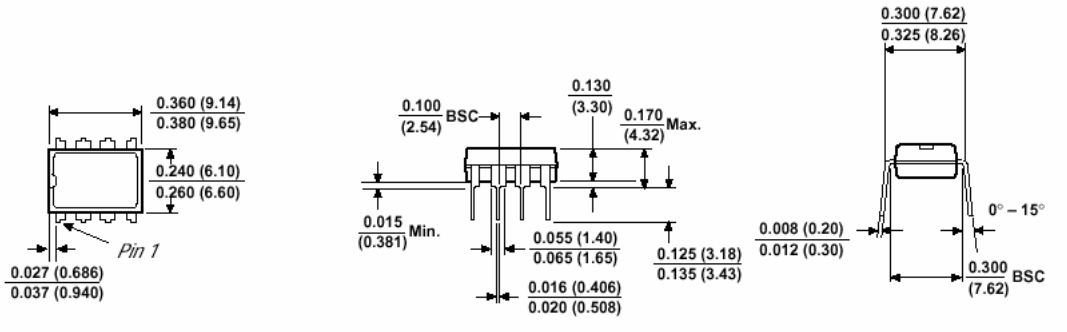
ТИПЫ КОРПУСОВ ПЛИС

Тип корпуса	Сокращение	Материал выводов	Покрытие выводов
Ceramic dual in-line	CerDIP	Сплав 42	Лужение
Plastic dual in-line	PDIP	Медь	Лужение
Ceramic J-lead chip carrier	JLCC	Сплав 42	Лужение
Plastic J-lead chip carrier	PLCC	Медь	Лужение
Ceramic pin-grid array	PGA	Сплав 42	Золото
Plastic small-outline integrated circuit	SOIC	Медь	Лужение
Ceramic quad flat pack	CQFP	Сплав 42	Олово, лужение
Plastic quad flat pack	PQFP	Медь	Лужение
Plastic thin quad flat pack	TQFP	Медь	Лужение
Power quad flat pack	RQFP	Медь	Лужение
Ball-grid array	BGA	Сплав олова (63/37)	-

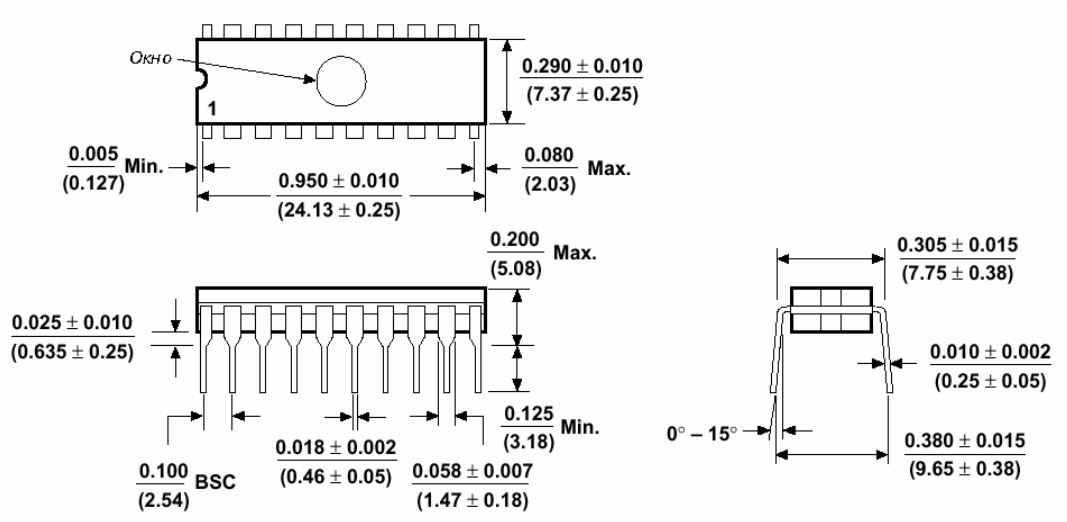
ПРИЛОЖЕНИЕ 3

ПРИМЕРЫ КОРПУСОВ

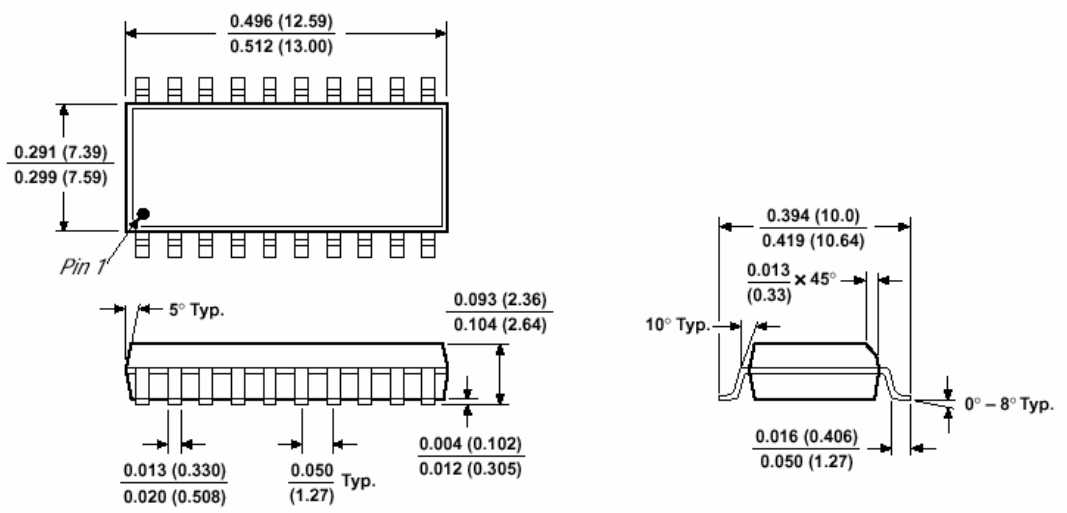
8-выводный корпус PDIP



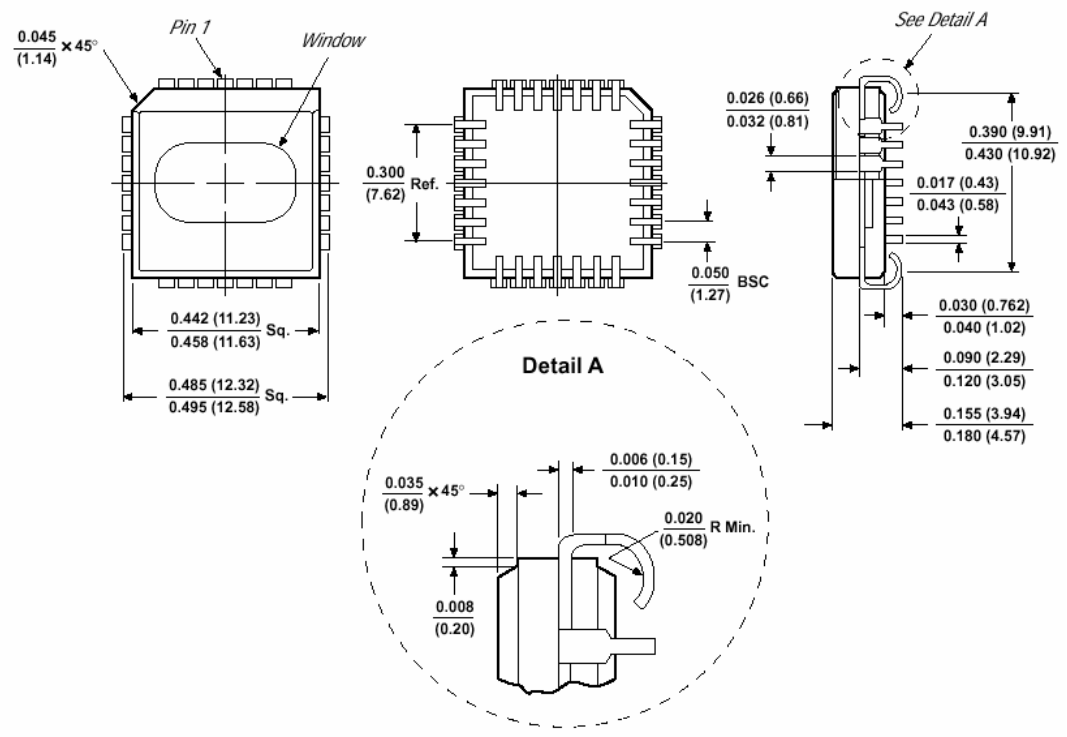
20-выводный корпус CerDIP



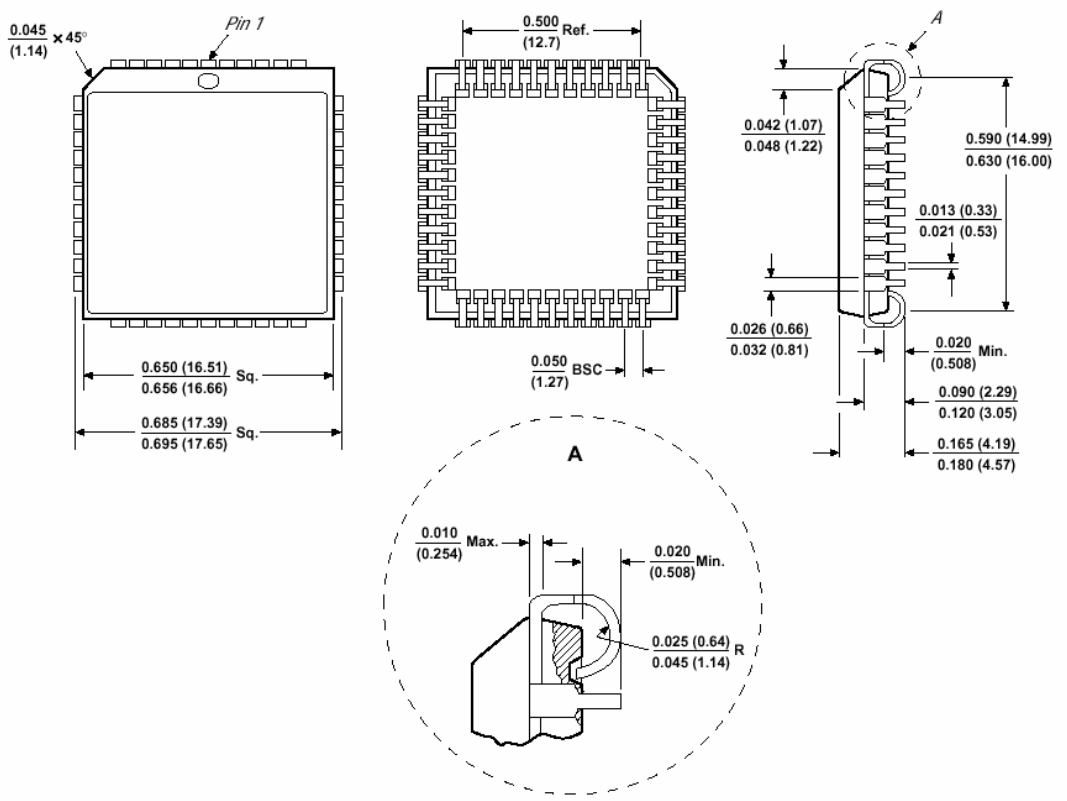
20-выводный корпус SOIC



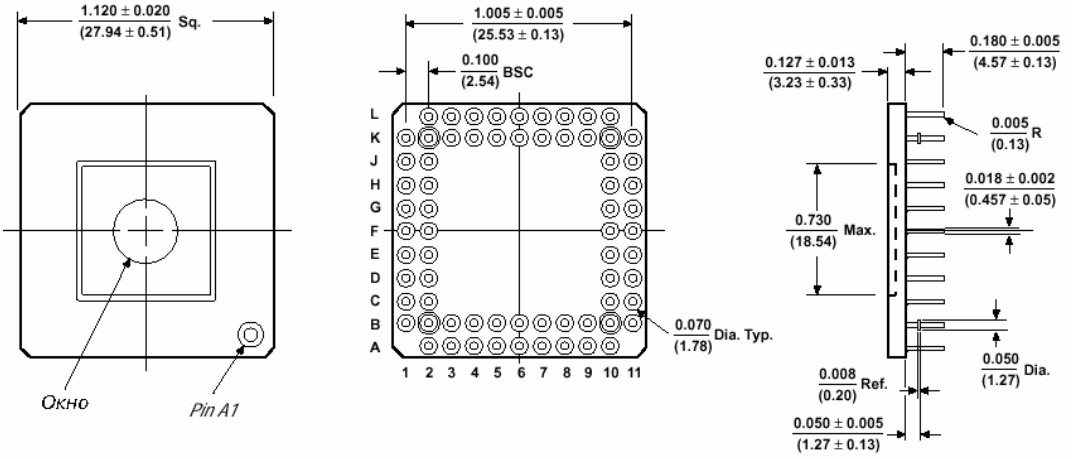
28-выводный корпус JLCC



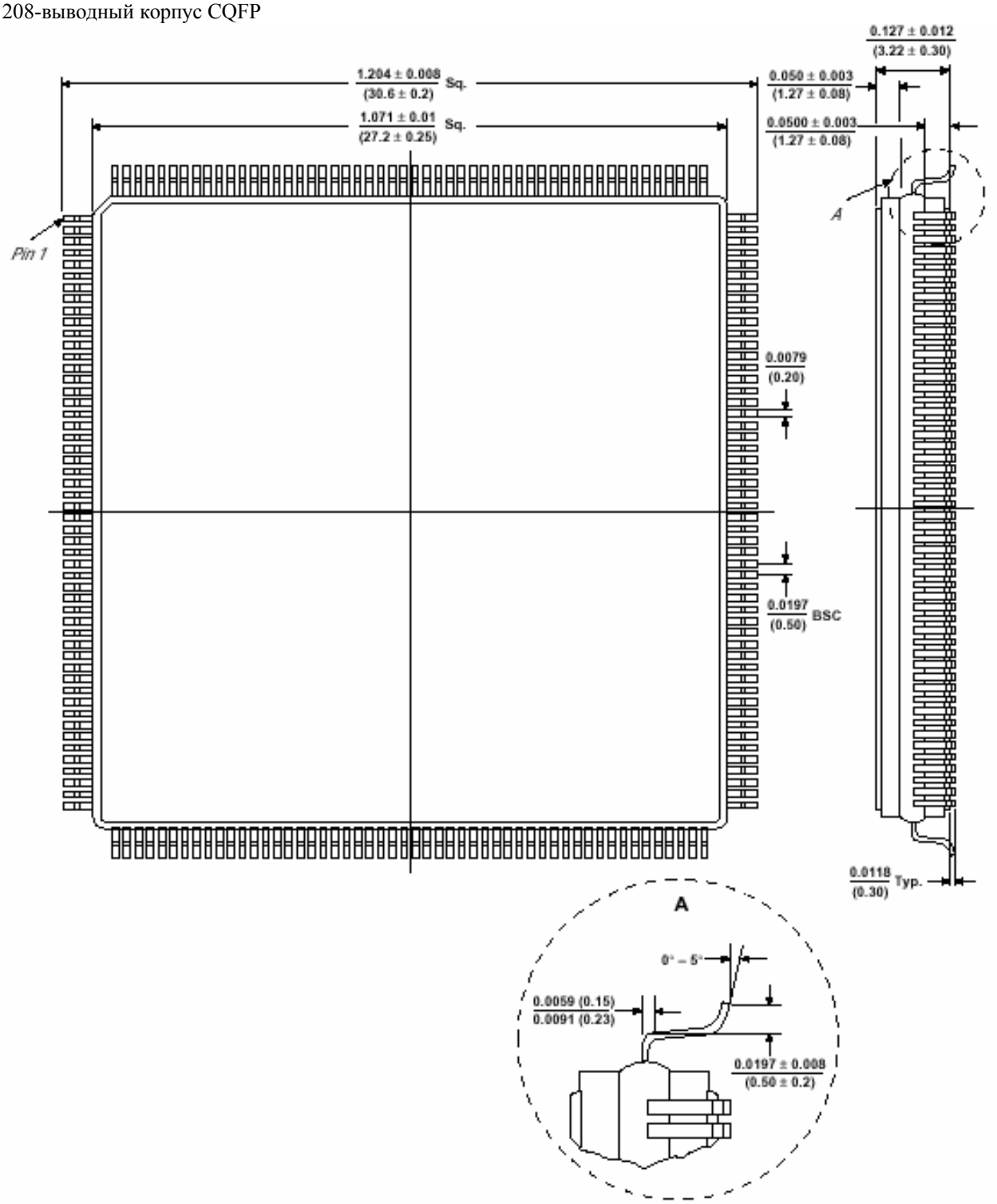
44-выводный корпус PLCC



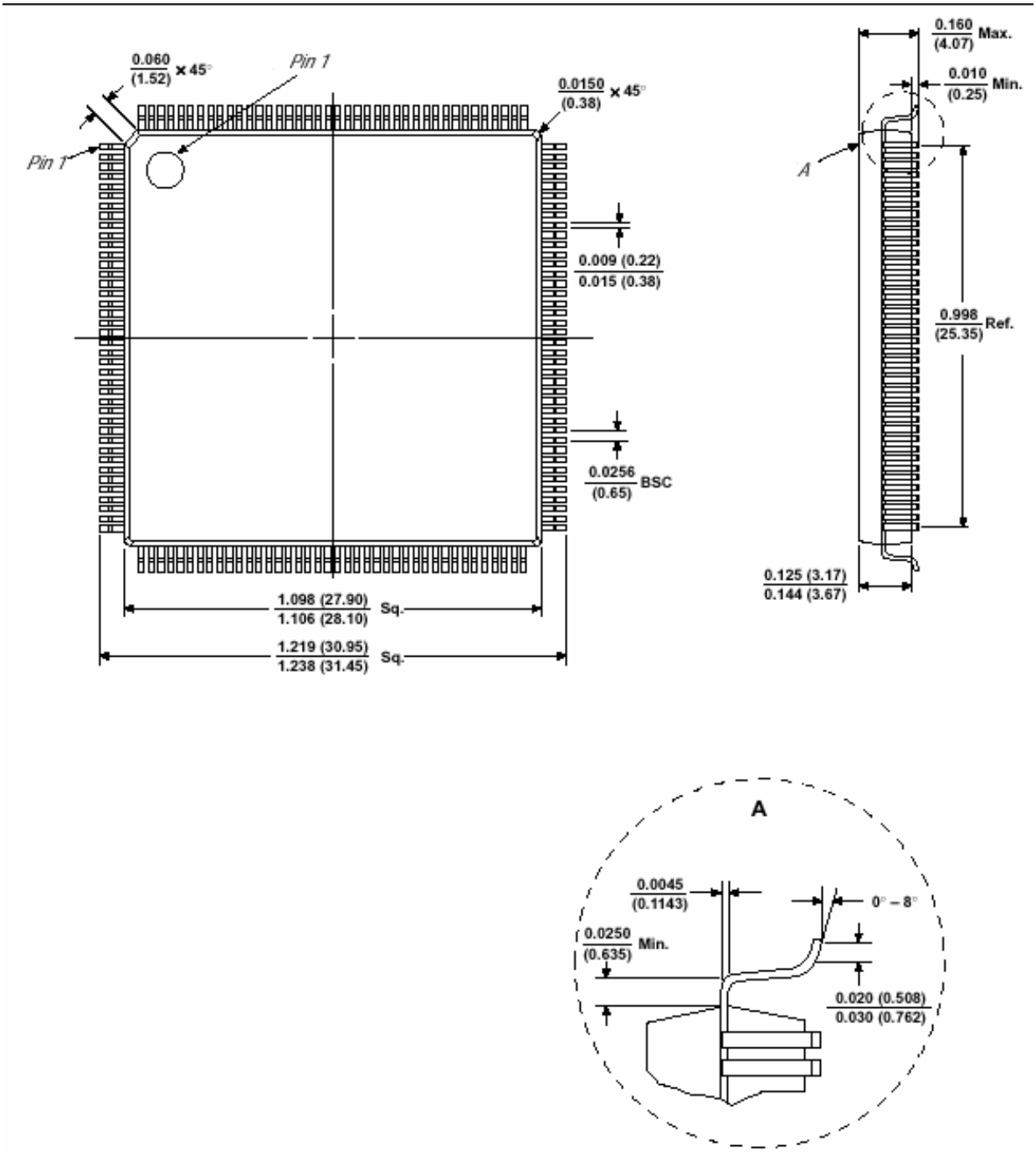
68-выводный корпус PGA



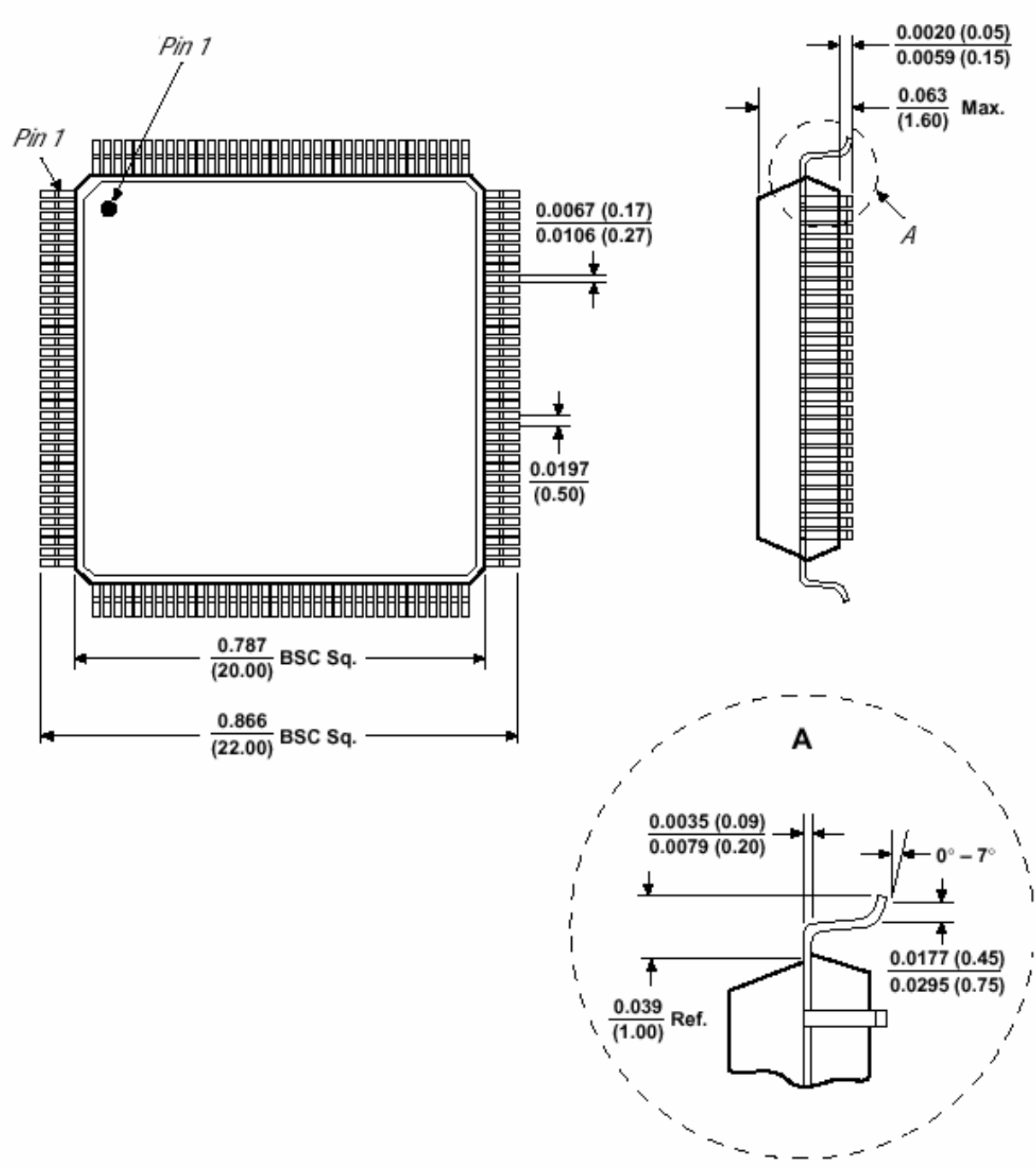
208-выводный корпус CQFP



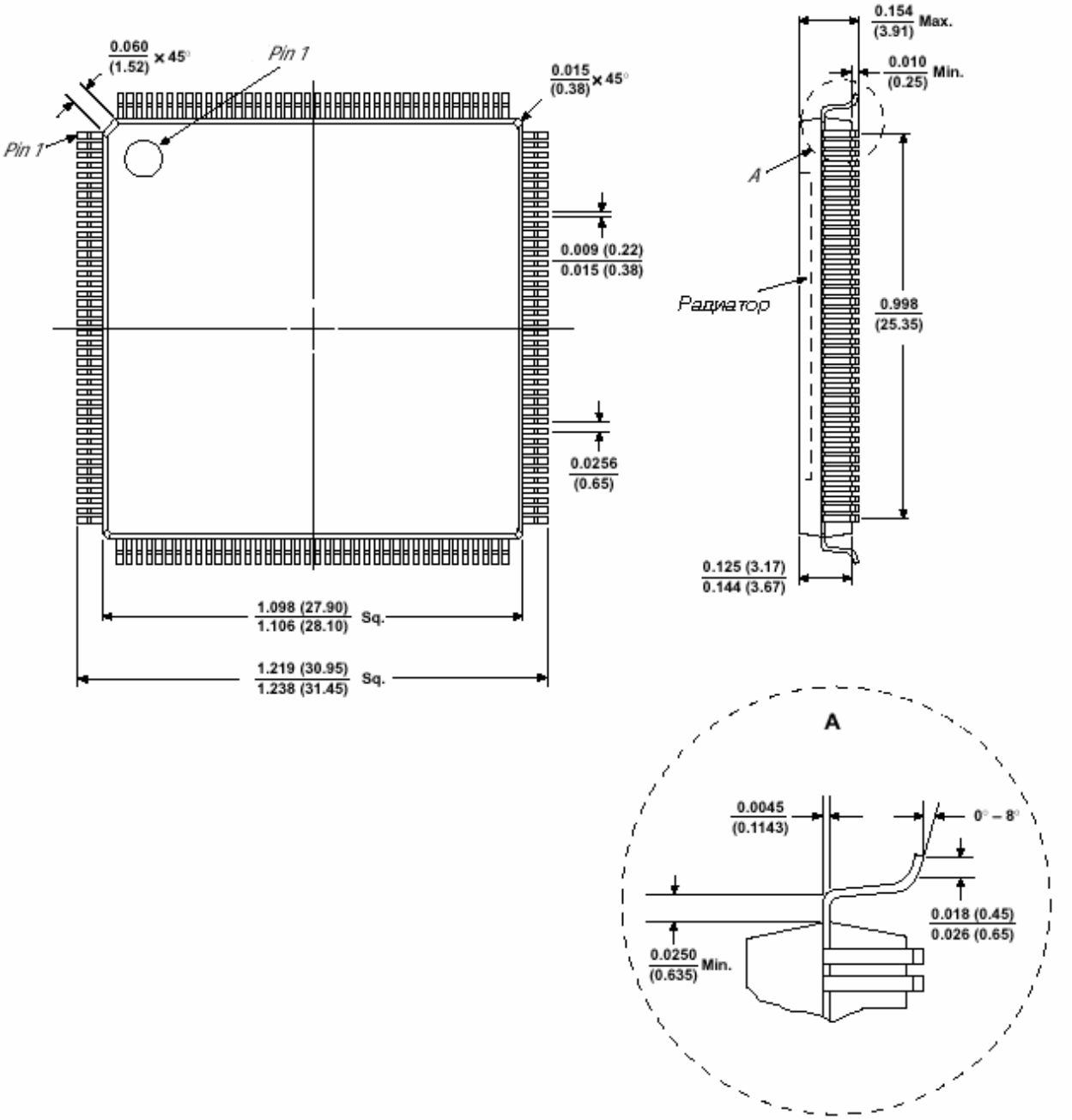
160-выводный корпус PQFP



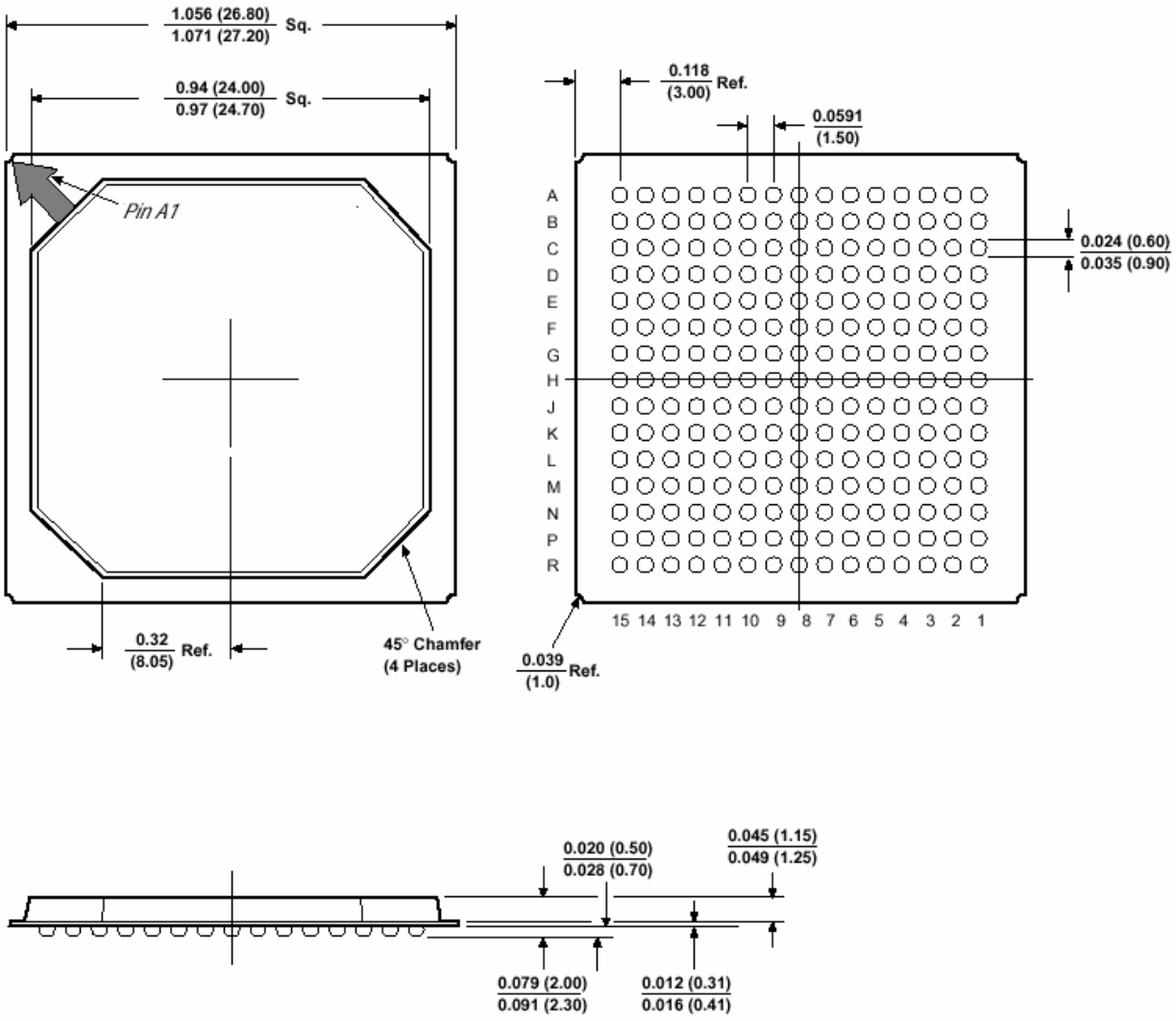
144-выводный корпус TQFP



160-выводный корпус RQFP



225-выводный корпус BGA



БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Баранов С.И., Скляров В.А. Цифровые устройства на программируемых БИС с матричной структурой.-М: Радио и связь, 1986.-272с.
2. Бадулин С.С., Барнаулов Ю.М. Автоматизированное проектирование цифровых устройств.- М: Радио и связь, 1981.-235с.

3. Data Book. - Altera Corporation, 1996.

4. The Programmable Logic Data Book. - Xilinx Inc., 1996.
5. MAX PLUS II. Getting Started. Version 6.0.
6. Антонов А.П., Мелехин В.Ф., Филиппов А.С. Обзор элементной базы фирмы Altera.-С.-Петербург, 1997.-142с.
7. Проектирование цифровых вычислительных машин/ Под ред. С.А. Майорова: Уч. пособие для студентов вузов.-М.:Высш. шк., 1972.-344с.
8. Потёмкин И.С. Функциональные узлы цифровой автоматики. –М.:Энергоатомиздат, 1988.-320с.
9. Лазер И.М., Шубарев В.А. Устойчивость цифровых микроэлектронных устройств.-М.:Радио и связь, 1983.-216с.
10. Левин В.И. Динамика логических устройств и систем.-М.:Энергия, 1980.-244с.
11. Воробьев Н. Методы анализа комбинационных схем на риски сбоя//CHIP NEWS. – 1996. - №3.
12. Армстронг Дж. Р. Моделирование цифровых систем на языке VHDL: /Пер. с англ. - М.: Мир, 1992. – 175 с.
13. Теория и техника радиосвязи: Науч.-техн. сб. Вып.1. - Воронеж, 1998.- 120 с.
14. Data Book. - Actel, 1995.
15. Flex 8000 Hand Book. - Altera Corporation, 1994.
16. Configurable Logic Data Book.- Atmel Corporation, 1997.

Учебное издание

Волков Александр Николаевич

Руфицкий Михаил Всеволодович

ПРОЕКТИРОВАНИЕ ЭЛЕКТРОННЫХ СРЕДСТВ
НА ОСНОВЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ
ИНТЕГРАЛЬНЫХ СХЕМ

Учебное пособие

Редактор Е.П. Викулова

Корректор В.В. Гурова

Изд. лиц. №020275. Подписано в печать 19.04.02.

Формат 60x84/16. Бумага для множит техники. Гарнитура Times.

Печать офсетная. Усл. печ. л. 6,51. Уч.-изд. л. 6,85. Тираж 100 экз.

Заказ

Редакционно – издательский комплекс

Владимирского университета.

600000, Владимир, ул. Горького, 87.