ГЛАВА 8

СХЕМОТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Давным-давно

Для начала рассмотрим способы проектирования цифровых микросхем, которые использовались давным-давно, т. е. в начале 60-х. Эта информация будет интересна читателю нетехнического склада ума, а также начинающим инженерам, которые достаточно хорошо разбираются в современных средствах и методах проектирования, но не знакомы с историей их возникновения и развития. Кроме того, все, о чем говорится здесь, помогает лучше понять суть и возможности современных методов проектирования, которые будут рассмотрены в последующих главах.

Давным-давно электронные схемы создавались вручную. Электрические схемы или *принципиальные схемы*, или просто *схемы*, вычерчивались вручную с помощью карандаша и трафарета, а иногда для этих целей использовалась скатерть, если это происходило в ресторане. На схемы наносились обозначения применяемых в устройстве логических вентилей и функций и изображались соединения между ними.

Соединения между логическими элементами внутри микросхемы могут называться *проводниками*, *дорожками* или *внутренними соединениями*, причём эти термины могут употребляться как синонимы. В некоторых случаях в этом же значении может использоваться термин *металлизация*, поскольку проводники в микросхемах преимущественно формируются в виде *слоёв металлизации*.

В каждой команде разработчиков был хотя бы один человек, превосходно выполняющий процедуры логической *минимизации* и *оптимизации*, которые сводились к замене одной группы логических элементов другой, выполняющей те же задачи, но быстрее или с использованием меньшего пространства на кремниевом кристалле.

Проверка функционирования конструкции на соответствие требуемому алгоритму работы, т. е. *функциональная проверка*, обычно выполнялась группой инженеров, которые сидели за столом и проверяли схему, приговаривая: «Да, вроде все правильно». Аналогично, верификация временных параметров, т. е. проверка на соответствие задержки сигнала от входов до выходов устройства и на внутренних блоках требуемым значениям, а также проверка на отсутствие нарушения временных ограничений, таких как время готовности и время занятости устройства, которые могут быть вызваны нарушением работы одного из внутренних регистров, выполнялась с помощью карандаша и листа бумаги. При этом в распоряжении самых удачливых были механические или электромеханические калькуляторы.

Затем вручную выполнялся набор чертежей, на которых были представлены элементы, используемые для формирования логических вентилей или, если быть более точным, транзисторы, формирующие логические вентили, и внутренние соединения между ними. Такие чертежи, состоящие из групп квадратов и прямоугольников, использовались для создания фотошаблонов, применяемых при изготовлении кремниевых кристаллов.

Развитие САПР электронных систем

1865 г. *Англия. Джеймс Клерк Максвелл (James Clerk Maxwell) предсказал существование электромагнитных волн, которые распространяются так же, как и свет.*

Начальный этап проектирования. Логическое моделирование

Неудивительно, что процесс разработки систем вручную, который рассматривался выше, был трудоемким и способствовал возникновению ошибок. В связи с этим необходимо было предпринять какие-то меры, и многие компании и университеты начали активно работать в различных направлениях. Так, например, для проведения функциональной верификации в конце 60-х — начале 70-х годов появились специальные программы в виде элементарных систем *логического моделирования*.

Чтобы понять принцип действия такой системы, представим, что в нашем распоряжении находится простое *устройство на логических вентилях*, принципиальная схема которого выполнена вручную на листе бумаги (**Рис. 8.1**).

Под *устройством на логических вентилях* подразумевается устройство, выполненное в виде набора логических вентилей и функций и соединений между ними. Чтобы использовать логическое моделирование, инженерам для начала необходимо было создать текстовое описание принципиальной схемы, которое называлось *таблицей соединения вентилей (gate-level netlist)*. Давным-давно разработчики обычно использовали мейнфраймы, т. е. большие ЭВМ общего назначения, и таблицы соединений, которые вводились с помощью набора перфокарт. Такой набор называли *колодой*. (Колода карт... Понятно, что к чему?) С внедрением персональных компьютеров вместе с устройствами хранения данных, таких как жесткие диски, таблицы соединений стали записывать в виде текстовых файлов (**Рис. 8.2**).

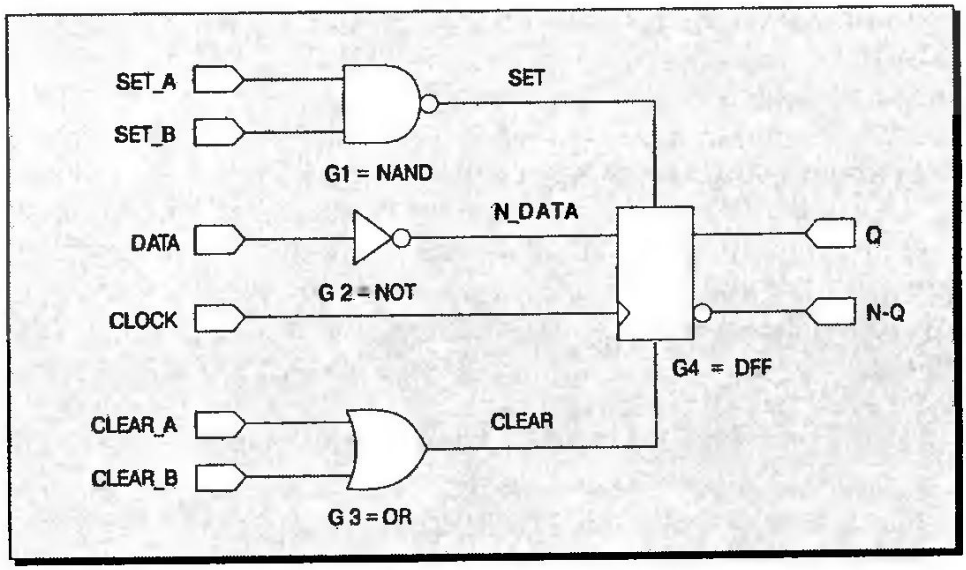


Рис. 8.1. Простая принципиальная схема, выполненная на бумаге

Существовала также возможность каждому вентилю сопоставить некоторую задержку распространения сигнала. Эти задержки, в примере они опущены для простоты изложения, обычно представлялись в виде чисел, кратных некоторой единице времени системы моделирования (*см.* гл. 19).

Форма представления таблицы соединений, показанная на **Рис. 8.2**, приведена исключительно в качестве примера. Она выполнена в духе того времени, когда, давным-давно исключительно ради под-

```
BEGIN    CIRCUIT=TEST
    INPUT     SET_A,     SET-B,     DATA,     CLOCK,     CLEAR_A,     CLEAR_B;
    OUTPUT    Q,   N_Q;
    WIRE      SET,    N_DATA,     CLEAR;

    GATE   G1=NAND    (IN1=SET_A,          IN2=SET_B,          OUT1=SET);
    GATE   G2=NOT     (IN1=DATA,          OUT1=N_DATA);
    GATE   G3=OR      (IN1=CLEAR_A,          IN2=CLEAR_B,          OUT1=CLEAR);
    GATE   G4=DFF     (IN1=SET,          IN2=N_DATA,          IN3=CLOCK,
                       IN4=CLEAR,          OUT1=Q,      OUT2=N_Q);

END   CIRCUIT=TEST;
```

Рис. 8.2. Простая таблица соединений вентилей — текстовый файл

держания хорошего настроения каждый разработчик системам моделирования стремился разработать собственный язык описания таблицы соединений.

Первые системы логического моделирования содержали в себе описания только примитивных вентилей, таких как И, И-НЕ, ИЛИ, ИЛИ-НЕ и др. Такие инструменты назывались *простейшими системами моделирования*. Позже некоторые системы стали поддерживать описания более сложных логических функций, таких как D-триггеры. В этом случае функция G4 = DFF, представленная на **Рис. 8.2**, реализуется с помощью собственного внутреннего описания.

В качестве альтернативного подхода можно было бы создать подсхему, назвать её DDF, описать её функциональность с помощью таблицы соединений примитивных логических вентилей И, И-НЕ и др. В этом случае функция G4 = DFF, представленная на **Рис. 8.2**, может быть реализована в системе моделирования в виде вызова обработчика описания этой подсхемы

На следующем этапе пользователи создавали набор *тестовых векторов*, или *задающих или внешних воздействий (stimulus)*, в виде наборов значений логических 0 и 1, которые предназначались для подачи на входы схемы. Эти тестовые векторы в текстовой форме и обычно представляли собой таблицу, (**Рис. 8.3**), причем любая информация после символа «;» воспринималась как комментарий.

При описании задающего воздействия в левой колонке указывалось его время действия. Для экономии места названия входных сигналов указывались вертикально.

Как следует из **Рис. 8.1** и **8.2**, между входом схемы DATA и D-триггером установлен инвертирующий логический элемент НЕ (NOT). Таким образом, в начальный момент времени на вход DATA подается логическая 1, которая затем инвертируется в логический 0, и это значение загружается в регистр по фронту синхроимпульса, т. е. при его переходе из 0 в 1, в момент времени 500. Аналогично в момент времени 1500 на вход DATA подается логический 0, который инвертором преобразуется в 1 и при значении времени 2000 по фронту синхроимпульса загружается в регистр.

В современной терминологии файл с тестовыми векторами, показанный на **Рис. 8.3**, представляет собой упрощенный *тестовый стенд*. Ещё раз повторюсь: значения времени обычно представлялись в виде чисел, кратных некоторой единице времени в системе моделирования.

```
                C C
                L L
          S S   C E E
          E E D L A A
          T T A O R R
          _ _ T C _ _
TIME      A B A K A B
-----     -----------
    0     1 1 1 0 0 0        ; Установка начальных значений
  500     1 1 1 1 0 0        ; Фронт синхроимпульса (загрузка 0)
 1000     1 1 1 0 0 0        ; Спад синхроимпульса
 1500     1 1 0 0 0 0        ; Установка данных в 0 (N_data = 1)
 2000     1 1 0 1 0 0        ; Фронт синхроимпульса (загрузка 1)
 2500     1 1 0 1 0 1        ; Очистка значения B (загрузка 0)
    :
  и так далее
```

Рис. 8.3. Простой набор тестовых векторов — текстовый файл

После этого инженеры запускали систему логического моделирования, которая считывала таблицу соединений вентилей и строила виртуальное представление принципиальной схемы в памяти компьютера. Затем система моделирования считывала первый тестовый вектор (первую строку из файла внешних воздействий), выставляла заданные значения на виртуальных входах и транслировала их через схему. Подобным образом процедура повторялась для каждого последующего тестового вектора, и, таким образом, формировалась работа тестового стенда (**Рис. 8.4**).

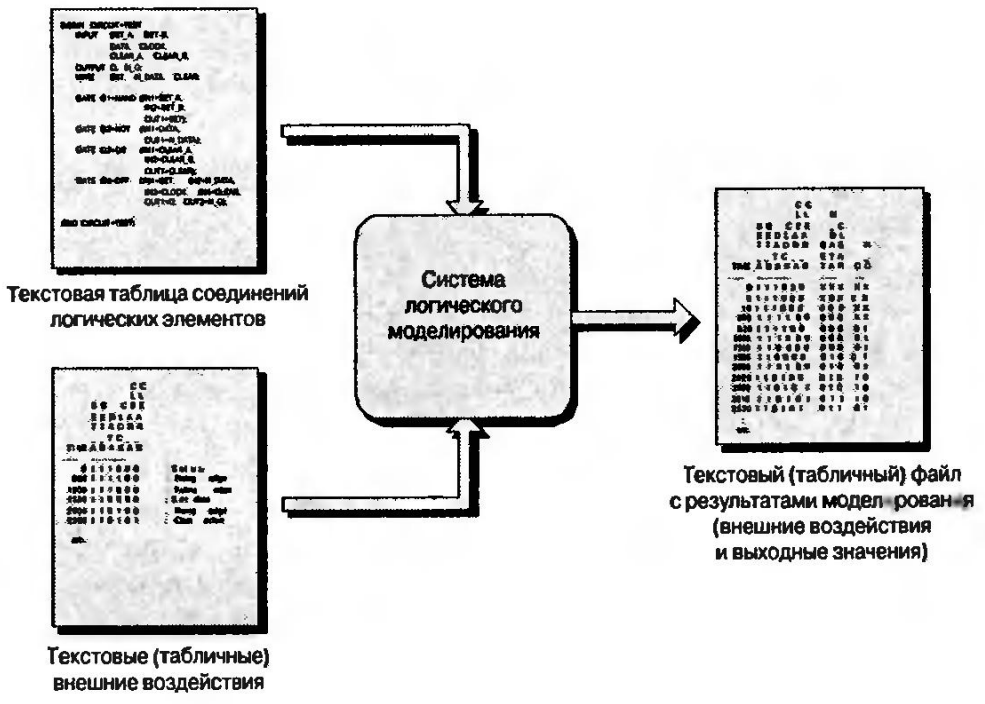


Рис. 8.4. Работа системы логического моделирования

Система моделирования может также использовать один или несколько управляющих файлов (или интерактивных команд) для обозначения внутренних узлов (проводников) и выходов схемы, которые подвергаются мониторингу в процессе моделирования, а также для

обозначения длительности процесса моделирования и установки других параметров. Результаты моделирования вместе с исходными воздействиями сохраняются в форме таблицы в текстовом файле.

Представим себе, что совершаем путешествие в прошлое и там запускаем одну из старых систем моделирования, используя описания схемы, изображённые на **Рис. 8.1** и **8.2** вместе с воздействиями, показанными на **Рис. 8.3**. Допустим, что инвертирующему логическому вентилю соответствует задержка сигнала величиной в пять временных единиц системы моделирования. Это значит, что после подачи сигнала на вход этого логического элемента, соответствующее ему значение на выходе появиться только через пять отсчётов времени. Аналогично, допустим, что логические элементы И-НЕ и ИЛИ имеют задержку в 10 временных единиц, а D-триггер задерживает сигнал на 20 отсчётов.

Если система моделирования была настроена на мониторинг всех внутренних узлов и выходных сигналов, выходной файл будет содержать результаты моделирования в виде, как показано на **Рис. 8.5**.

```
                C C
                L L     N
      S S   C E E       _ C
      E E D L A A       D L
      T T A O R R     S A E     N
      _ _ T C _ _     E T A     _
 TIME A B A K A B     T A R   Q Q
-----  -----------    -----   ---
    0 1 1 1 0 0 0    X X X   X X  ; Установка начальных значений
    5 1 1 1 0 0 0    X 0 X   X X
   10 1 1 1 0 0 0     0 0 0  X X

  500 1 1 1 1 0 0    0 0 0   X X  ; Фронт синхроимпульса
  520 1 1 1 1 0 0    0 0 0    0 1

 1000 1 1 1 0 0 0    0 0 0   0 1  ; Спад синхроимпульса

 1500 1 1 0 0 0 0    0 0 0   0 1  ; Установка данных в 0
 1505 1 1 0 0 0 0    0 1 0   0 1

 2000 1 1 0 1 0 0    0 1 0   0 1  ; Фронт синхроимпульса
 2020 1 1 0 1 0 0    0 1 0    1 0

 2500 1 1 0 1 0 1     0 1 0  1 0  ; Активный сигнал Clear_B
 2510 1 1 0 1 0 1    0 1 1   1 0
 2530 1 1 0 1 0 1    0 1 1    0 1
   :
 и так далее
```

Рис. 8.5. Результаты моделирования (текстовый файл)

Для улучшения восприятия нашего примера все изменения значений сигналов на рисунке выделены жирным шрифтом, хотя на практике этот прием не применялся.

Начальные значения воздействия подаются на входы схемы в момент времени 0. В этот момент времени состояния всех внутренних узлов и выходов в таблице обозначены значением *X*, что означает неопределённое состояние. По прошествии пяти единиц времени, начальное значение логической 1, которое было подано на вход DATA, проходит через инвертирующий логический элемент НЕ и преобразуется в логический 0 на внутреннем узле N_DATA. Аналогично по про-

шествии 10 единиц времени, начальные значения, которые были представлены на входы схемы SET_A и SET_B, проходят через логический элемент И-НЕ на внутренний узел SET. Одновременно сигналы, поданные на входы CLEAR_A и CLEAR_B, проходят через логический элемент ИЛИ на внутренний узел CLEAR.

В момент времени 500 по фронту синхроимпульса на входе CLOCK D-триггер загружает значение, находящееся в этот момент на внутреннем узле N_DATA. Спустя 20 временных единиц это значение появляется на выходах Q и N_Q триггера. И так далее.

Пустые строки в выходном файле, такие как между записями, соответствующие значениям времени 10 и 500, используются для обозначения родственных групп действий. Например, установка начальных значений в момент времени 0 является причиной изменения сигналов в моменты времени 5 и 10. Затем фронт синхроимпульса на входе CLOCK в момент времени 500 вызывает изменение сигналов при достижении временной отметки 520. Так как эти две группы действий абсолютно не зависят друг от друга, то они разделяются пустой строкой.

Всё вышесказанное происходило незадолго до того, как инженеры начали работать со схемами, содержащими тысячи вентилей и узлов, моделирование которых проводилось на протяжении нескольких тысяч временных шагов (тактов). Да, я часами просиживал, анализируя файлы, подобные только что рассмотренным, пытаясь понять, работает ли система так, как ей положено, и отчаянно пытаясь найти ошибку, если что-то было не так.

Завершающий этап проектирования. Компоновка

Инструменты, подобные системам логического моделирования, были призваны помочь инженерам определить функциональность микросхемы и печатной платы. В то же время некоторые компании сосредоточили свои усилия на создании средств проектирования, которые помогали бы выполнять компоновку внутренних узлов микросхемы. В этом контексте термин *компоновка* имеет отношение и к расположению вентилей (на самом деле транзисторов, из которых состоит вентиль) на поверхности кристалла, и к тому, как прокладывать соединения, связывающие их между собой.

В начале 70-х такие компании, как Calma, ComputerVision и Applicon создали специальные компьютерные программы, которые позволяли сотрудникам конструкторских отделов переводить разработанные вручную схемы в цифровой вид. Для этого схема устройства помещалась на широкий цифровой планшет графического ввода. Затем с помощью устройства, напоминающего компьютерную мышку, выполнялась оцифровка границ фигур, т. е. многоугольников, используемых для определения транзисторов и внутренних соединений. Цифровые файлы, созданные таким способом, использовались для создания фотошаблонов, которые, в свою очередь, применялись при производстве кремниевых кристаллов.

Спустя некоторое время эти первые компьютерные инструменты для конструкторов развились до уровня интерактивных программ, называемых *редакторами многоугольников*, которые позволяли пользователям рисовать многоугольники непосредственно на экране компьютера. Потомки этих программ со временем получили возможность использовать применяемые в системах моделирования таблицы соединений логических элементов, по которым автоматически выполнялись задачи разводки микросхем.

1865 г. *Атлантический кабель связал Валенсию (Ирландия) и Тринити Бей (Ньюфаундленд).*

1866 г. *Ирландия/США. Проложен первый долговременный трансатлантический кабель.*

1869 г. *Вильям Стэнли Джеванс (William Stenly Jevans) изобрёл «логическое пианино».*

1872 г. *Осуществлена первая одновременная передача с двух концов телеграфного провода.*

САПР электронных систем

Средства проектирования, такие как системы логического моделирования, которые использовались на первых этапах проектирования, т. е. на этапах описания принципиальной схемы и функциональной проверки, изначально имели общее название *системы автоматизированной разработки (моделирования) — CAE (computer-aided engineering)*. А инструменты компоновки, т. е. размещения элементов и трассировки, использующиеся на завершающих, или физических, этапах, изначально имели общее название *системы автоматизированного проектирования (конструирования) — CAD (computer-aided design)*.

Исторически сложилось так, что, в основном основываясь на определении понятий систем автоматического конструирования и проектирования, термин *проектировщик,* или просто *инженер,* обычно относился к человеку, занятому на первых этапах разработки, т. е. выполняющему задачи по составлению и описанию, или вводу, функциональности микросхемы, т. е. что она делает и как она это делает. В свою очередь, понятие *инженер-конструктор,* или просто *конструктор,* в общем случае относилось к человеку, который работает на завершающем этапе проектирования, т. е. на этапе конструирования, и выполняет задачи по разводке узлов микросхемы, определения расположения логических элементов и их связи между собой.

В 80-х все системы автоматизированного конструирования и проектирования электронных компонентов и систем получили одно название — *системы автоматизации проектирования электронных приборов и устройств*, или сокращенно *САПР электронных устройств (EDA — electronic design automation)*. Таким решением, казалось, довольны были все. Однако были, все-таки, недовольные, но никто не обращал внимания на их недовольные выкрики...

Термин САПР также используется во многих других областях, например в машиностроительном конструировании и архитектуре.

Простой метод схемотехнического проектирования заказных ИС

В конце 70-х и начале 80-х компании Daisy, Mentor и Valid выпустили первые программы *графического описания схем*, которые позволяли инженерам интерактивно создавать принципиальные схемы непосредственно на компьютере. С помощью компьютерной мышки инженер мог выбрать из *специальной библиотеки элементов* графическое обозначение различных элементов, например контакты ввода/вывода, логические вентили или функциональные узлы, и поместить их в нужном месте на экране компьютера. Позже у инженеров появилась возможность с помощью мышки чертить линии или проводники, соединяя вместе различные элементы.

После ввода в компьютер описания принципиальной схемы программа по указанию пользователя могла генерировать соответствующую таблицу соединений логических элементов. Эта таблица могла быть использована, во-первых, в качестве исходных данных для системы логического моделирования, т. е. для проверки функциональности устройства, и, во-вторых, для работы программного обеспечения по размещению элементов и трассировки соединений (**Рис. 8.6**).

Программа производила оценку всех временных параметров, которые в начале использовались системой логического моделирования. Особое внимание уделялось внутренним проводникам, так как возможность провести точный временной анализ появлялась только пос-

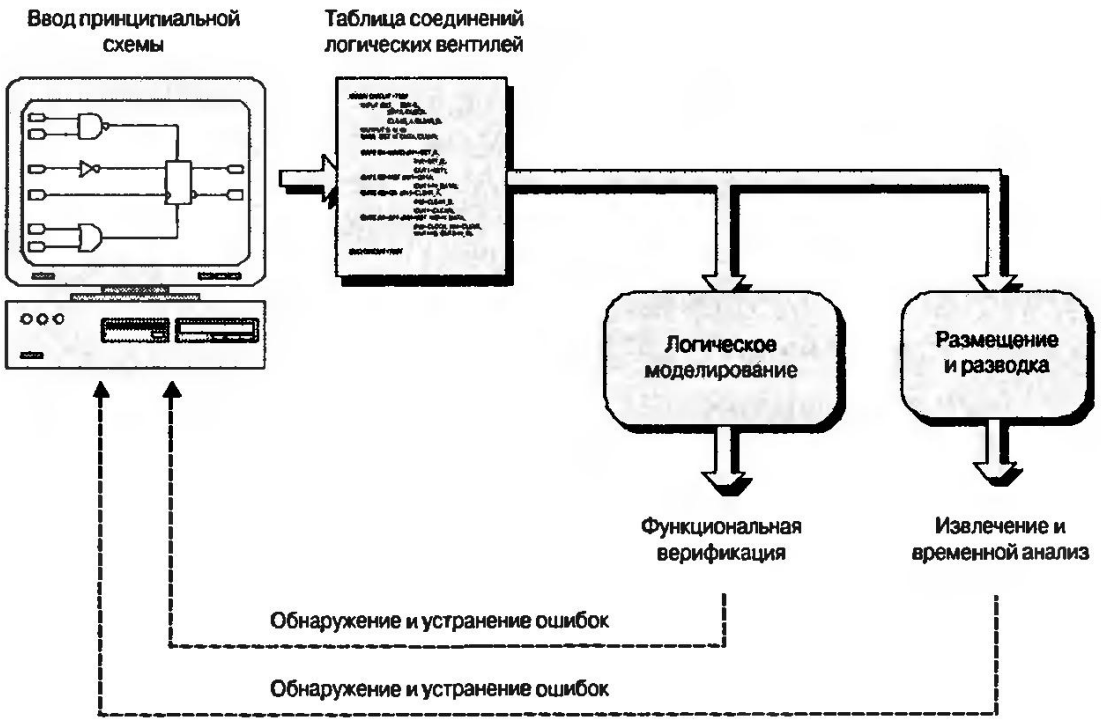


Рис. 8.6. Простой цикл схемотехнического проектирования заказных микросхем

1873 г. *Англия. Джеймс Клерк Максвелл (James Clerk Maxwell) описал электромагнитную природу света и опубликовал свою теорию электромагнитных волн.*

ле размещения всех логических элементов и разводки внутренних связей между ними. Другими словами, после процедуры размещения элементов и их разводки специальная программа извлекала, т. е. выделяла, и рассчитывала величины паразитных сопротивлений и ёмкостей, связанных с элементами, формирующими схему: сегментами проводников, переходными отверстиями транзисторами и так далее. Затем эти данные использовались программой временного анализа для создания отчёта о временных параметрах устройства. В некоторых системах проектирования эта временная информация также возвращалась в подсистему логического моделирования для проведения более точного моделирования.

Особого внимания заслуживает тот факт, что при создании схемы устройства пользователю через условные обозначения были доступны логические вентили и функции из специальной библиотеки, которая связана с определённой технологией изготовления специализированных микросхем[1]. Также система моделирования могла быть настроена на использование определенной библиотеки моделей с требуемой логической функциональностью[2] и временными параметрами, соот-

---

[1] Существуют различные способы построения библиотек. Например, некоторые их виды основаны на концепции использования наборов элементов общего назначения, которые входят в состав большинства библиотек заказных ИС. В этом случае таблица соединений, сгенерированная после создания принципиальной схемы устройства, пропускалась через транслятор, который производил преобразование наименований из библиотеки общего назначения в эквиваленты, используемые библиотекой конкретной заказной микросхемы.

[2] Что касается функциональности, то для многих она ассоциируется с простейшим логическим объектом, например 2-входовым элементом И, который функционирует одинаково на большинстве платформ. Конечно, такая однозначность будет только в том случае, когда на входы этих элементов подаются «правильные» значения (логические 0 и 1), ситуация сильно меняется, если на входы подаётся высокоимпедансное состояние «Z» или неизвестное значение «X». Даже при подаче на входы логических 0 и 1, более сложные функции, такие, как защелки и триггеры, могут вести себя непредсказуемо в необычных ситуациях, например при одновременной активации входов «установка» и «сброс».

1874 г. *Америка. Александр Белл (Alexander Graham Bell) разработал идею телефона.*

1875 г. *Америка. Эдисон изобрёл мимеограф.*

ветствующими определенной технологии изготовления заказных микросхем. В результате работы рассмотренных систем получалась таблица соединений, которая передавалась компоновщику-трассировщику для размещения и разводки. Таблица однозначно определяла структуру построения логических вентилей и функций на поверхности кристалла. В этом и заключается небольшое отличие в подходах к проектированию ASIC и ПЛИС.

Простой метод схемотехнического проектирования ПЛИС

С появлением первых ПЛИС в 1984 году по вполне естественным причинам их технологии проектирования были основаны на существующих схематических подходах проектирования заказных микросхем. Несомненно, сходство первых этапов проектирования заключалось в том, что программный блок описания схемотехнических изображений использовался для представления устройства в виде набора простейших логических вентилей и функций, а также для создания соответствующей таблицы соединений. Как и раньше, эта таблица использовалась системой моделирования для функциональной верификации устройства.

Различие начиналось на этапе реализации устройства, так как структура ПЛИС состояла из массива *конфигурируемых логических блоков (КЛБ)*, каждый из которых формировался с помощью таблиц соответствия и регистров. Это потребовало введения в процесс проектирования некоторых дополнительных шагов, названных *сопоставлением* и *компоновкой* (**Рис. 8.7**).

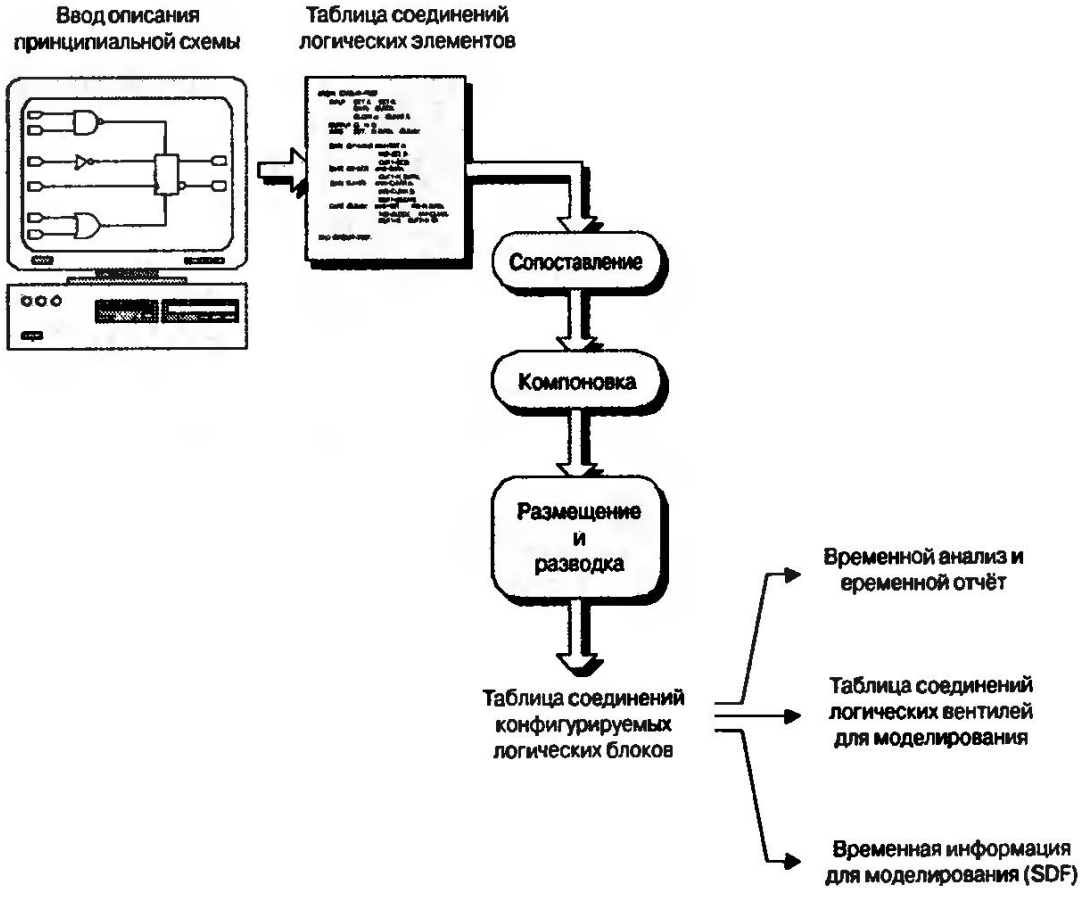
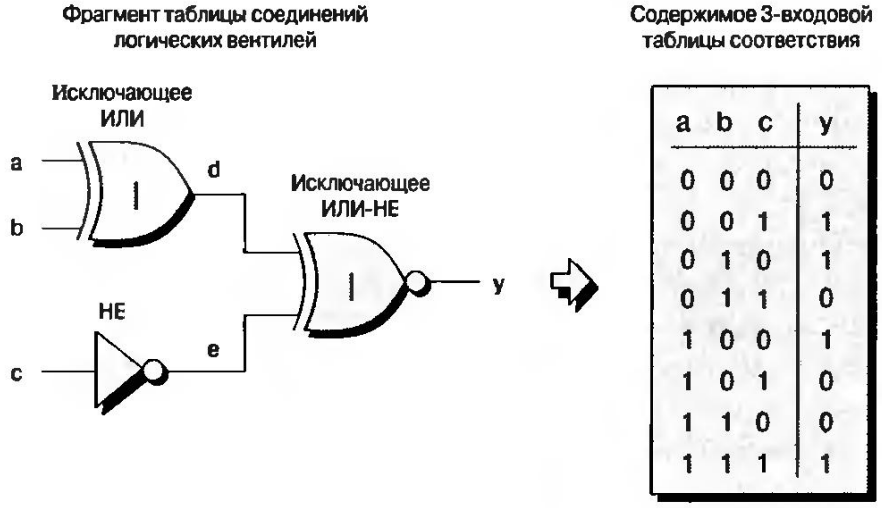


Рис. 8.7. Простой цикл схемотехнического проектирования ПЛИС

Сопоставление

1875 г. *Англия. Джеймс Максвелл (James Clerk Maxwell) установил, что атомы должны иметь определённую структуру.*

В контексте рассматриваемого материала *сопоставление* означает процесс установки соответствия между объектами, например между логическими функциями из таблицы соединений логических вентилей и таблицами соответствия ПЛИС. Конечно, речь не идёт о сопоставлении по принципу один к одному, так как каждая таблица соответствия может использоваться для представления нескольких логических элементов (**Рис. 8.8**).



a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Рис. 8.8. Сопоставление логических вентилей с таблицей соответствия

Сопоставление проводится и в наше время, но, как будет показано, на другом этапе проектирования ПЛИС, и является нетривиальной задачей, поскольку имеется много вариантов разбиения вентилей, формирующих таблицы соединений, на небольшие группы для их последующей реализации с помощью таблиц соответствия. Например, логический вентиль НЕ, показанный на **Рис. 8.8**, может реализовываться не в представленной таблице соответствия, а в предыдущей, к выходу которой подключается вход *c*.

Компоновка

После завершения этапа сопоставления наступает очередь этапа *компоновки*, в процессе которого таблицы соответствия и регистры распределяются по конфигурируемым логическим блокам (КЛБ). Процесс компоновки (который проводится и в наши дни, но, как будет показано, на другом этапе проектирования ПЛИС) также нетривиальная задача, поскольку имеется множество вариантов перестановок и сочетаний элементов логических ячеек по логическим блокам. В качестве примера рассмотрим простейшее устройство, состоящее из нескольких логических вентилей, которые на предыдущем этапе были сопоставлены с четырьмя 3-входовыми таблицами соответствия, обозначенными символами *A*, *B*, *C* и *D*. Допустим, что необходимо реализовать устройство на ПЛИС, в котором каждый конфигурируемый логический блок состоит из двух 3-входовых таблиц соответствия. В этом случае нам потребуется два логических блока (назовём их 1 и 2). На первый взгляд, существует 4! (факториал четырёх = $4 \times 3 \times 2 \times 1 = 24$) различных способа распределения наших таблиц по двум логическим блокам (**Рис. 8.9**).

Рис. 8.9. Разбивка таблиц соответствия по логическим блокам

На **Рис. 8.9** показаны только 12 из 24 возможных перестановок (читателю предлагается потренироваться и дописать недостающие варианты). На самом деле существует только 12 вариантов перестановок, имеющих практический смысл. Объясняется это тем, что существуют так называемые «зеркальные» перестановки, которые функционально эквивалентны, например пары *AC-BD* и *BD-AC*, показанные на **Рис. 8.9**. Эквивалентность этих пар связана с работой системы размещения и разводки, которая позволяет менять расположение двух логических блоков между собой.

Размещение и разводка

Двигаясь дальше, приходим к этапу *размещения элементов и разводки или трассировки соединений*. Вернемся к только что рассмотренному примеру и предположим, что два логических блока необходимо соединить вместе, но применительно к нашему случаю, они могут быть расположены только так, чтобы один из них являлся смежным по горизонтали или по вертикали с другим блоком. С учётом этих ограничений конфигурируемые логические блоки могут быть расположены по одному из четырёх вариантов (**Рис. 8.10**).

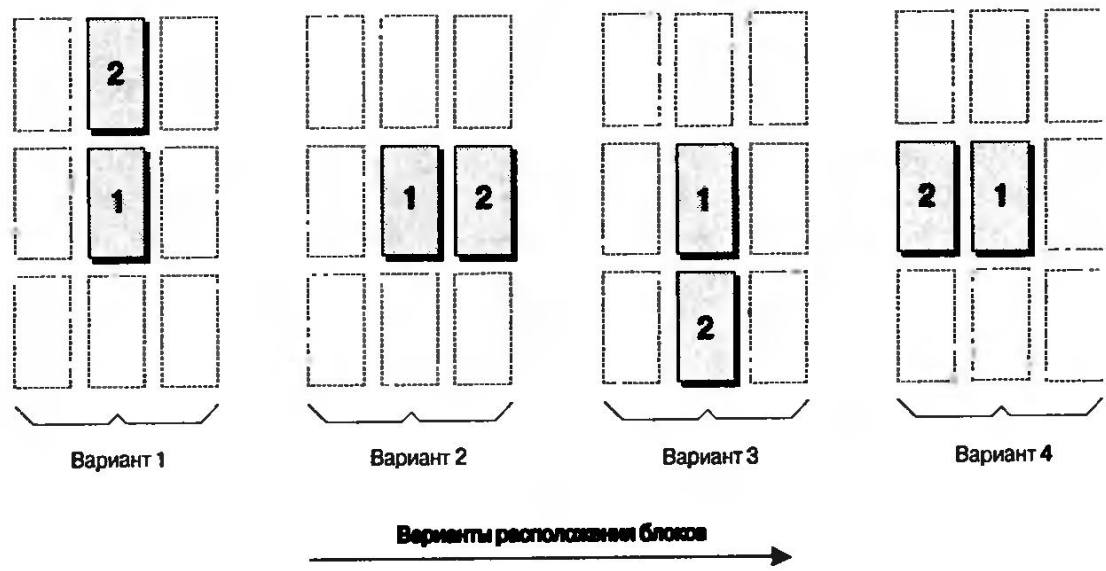


Рис. 8.10. Расположение конфигурируемых логических блоков

Если КЛБ 1 содержит таблицы соответствия *A* и *C*, а КЛБ 2 таблицы соответствия *B* и *D*, можно получить эквивалентное расположение. Для этого надо чтобы блоки поменялись местами и обменялись своим содержимым.

При работе с двумя логическими блоками **(Рис. 8.10)** довольно просто определить их оптимальное расположение относительно друг друга, т. е. выбрать наиболее оптимальный вариант из четырех возможных, а также относительно других элементов на кристалле.

На практике размещение элементов является куда более сложной задачей, чем рассматривалось в примере, поскольку реальные микросхемы могут содержать чрезвычайно большое количество конфигурируемых логических блоков: от нескольких сотен, на заре своего развития, и до нескольких сотен тысяч по состоянию на 2004 год. Соединенные вместе логические блоки 1 и 2 также могут подключаться к другим блокам. Например, КЛБ 1 может подключаться к КЛБ 3, 5 и 8, а КЛБ 2 может подключаться к КЛБ 4, 6, 7 и 8. Кроме того, эти блоки могут соединяться друг с другом и с другими блоками. Другими словами, хотя расположение КЛБ 1 и КЛБ 2 в непосредственной близости друг к другу может оказаться очень эффективным с точки зрения разводки соединений между ними, но с учётом их взаимосвязи с другими блоками такая расстановка может быть нежелательной. В некоторых случаях наиболее оптимальным может оказаться расположение, при котором КЛБ 1 и КЛБ 2 находятся на некотором расстоянии один от другого.

Если и размещение логических блоков является довольно сложной процедурой, оптимальная разводка соединений между этими блоками оказывается куда более сложной задачи. Сложность этих операций настолько огромна, что здесь лучше предоставить слово ребятам, безусловно, очень умным людям, которые заняты разработкой алгоритмов расположения элементов и разводки соединений, а мы перейдём к рассмотрению других вопросов.

Ещё до появления микросхем ПЛИС в мире ПЛУ такой же функциональностью, как и у утилит размещения и разводки, обладали приложения под названием «сборщики». С появлением ПЛИС название «сборщик» (fitter) также перешло и в эту сферу, однако со временем оно вытеснилось понятием «размещение и разводка» (place-and-route), которое более точно отражало суть выполняемых им действий.

В начале 90-х гг. некоторые поставщики вместо символической библиотеки простейших логических элементов и регистров стали использовать символические библиотеки более сложных функций, которых насчитывалось около 70. После разработки схемы формировалась таблица соединений функциональных блоков, которые, как правило, уже были сопоставлены таблицам соответствия и скомпонованы в КЛБ. Этот подход позволял лучше реализовать идею считать уровни логики между регистрами, но ограничивал возможности по оптимизации и замене элементов.

Временной анализ и повторное моделирование

Результатом процедуры размещения и разводки элементов является полная таблица физических соединений (на уровне КЛБ), как показано на **Рис. 8.7**. С помощью утилиты статического временного анализа *(STA — static timing analysis)* выполняется расчет значений всех временных задержек как на внутренних участках, так и при прохождении сигнала от входа до выхода микросхемы, а также проверка всех временных параметров, т. е. времени готовности, времени занятости и других, связанных с работой регистров.

Интересная ситуация складывается, когда разработчики решают произвести повторное моделирование своего устройства с учётом точной, т. е. полученной после процедуры размещения и разводки элементов, временной информации. В этом случае они используют комп-

1876 г. *Америка. 10-е марта. Александр Белл (Alexander Graham Bell) впервые осуществил телефонную связь.*

лект ПЛИС-утилит для создания новой таблицы соединений логических вентилей (с учётом временной информации) в виде файла промышленного формата, называемого, как это не удивительно, *форматом стандартных задержек (SDF — standard delay format)*. Необходимость создания этого файла обусловлена тем, что при обратном переходе от таблицы соединения, приведенной к уровню конфигурируемых логических блоков (КЛБ), к исходной таблице соединения логических вентилей просто невозможно соотнести новую и полученную ранее временную информацию.

Одноуровневые и иерархические принципиальные схемы

Одноуровневые принципиальные схемы

Первые программы ввода и принципиальных схем позволяли описывать огромные одноуровневые принципиальные схемы посредством их разделения на некоторое количество «страниц». Допустим, вы хотите нарисовать на листе бумаги принципиальную схему, состоящую из 1000 логических вентилей. Для полного изображения этой схемы потребуется огромный лист бумаги (около одного квадратного метра). На левой стороне листа будут отображены входы схемы, справа выходы, а между ними сама схема.

В таком виде схему крайне неудобно переносить для демонстрации из одного места в другое. Но можно разрезать на несколько страниц, которые будут храниться вместе в одной папке. В этом случае следует произвести разделение схемы таким образом, чтобы каждая страница содержала логические вентили, относящиеся к определенной функции системы. Также следует использовать межстраничные соединения, подобные входам и выходам, для прохождения сигналов между различными страницами.

По такому же принципу работали первые программы ввода принципиальных схем. Разработчики создавали одноуровневую принципиальную схему на нескольких страницах, связанных вместе с помощью межстраничных связей. Их названия указывали системе на порядок соединения страниц. Рассмотрим подобную простую схему, выполненную на листе бумаги (**Рис. 8.11**).

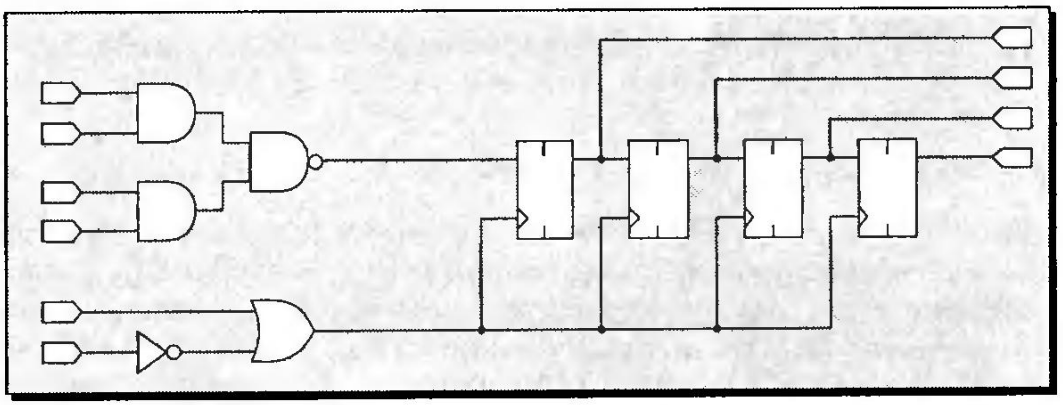


Рис. 8.11. Простая принципиальная схема, выполненная на листе бумаги

Допустим, что логические вентили в левой части схемы представляют собой некую схему управления, а в правой части четыре регистра реализуют 4-битный сдвиговый регистр. Очевидно, что это очень простой пример, реальные принципиальные схемы состоят из гораз-

до большего количества логических вентилей. Данный пример — это всего лишь попытка продемонстрировать принцип, согласно которому принципиальную схему при её описании можно разбить на две части (**Рис. 8.12**).

1876 г. *Александр Белл получил патент на телефон.*

Рис. 8.12. Простая двухстраничная одноуровневая принципиальная схема

Иерархические принципиальные схемы

При использовании одноуровневых принципиальных схем, особенно если эта схема занимает более 50-ти страниц, возникали следующие проблемы:

- Было трудно визуально представить все устройство как единое целое.
- Довольно сложно было повторно использовать части устройства в новых проектах.
- Если какая-либо часть схемы встречалась в устройстве несколько раз, что, кстати, случалось нередко, копию приходилось перерисовывать на нескольких страницах. Проблема усугублялась при необходимости внести некоторые изменения в эту часть схемы, т. е. приходилось перерисовывать все копии.

Решение этих проблем предусматривало расширение пакета программ ввода и описания принципиальных схем, в результате чего появлялась возможность реализации иерархической концепции. Например, при реализации рассмотренной выше схемы сдвигового регистра создается базовая страница, на которой будут изображены два блока: управления и сдвига. Каждый блок должен быть оснащен необходимыми входными и выходными контактами. Эти блоки должны соединяться межу собой, а также с входами и выходами устройства.

Затем можно перейти к блоку управления, при выборе которого откроется новая страница для ввода принципиальной схемы. Если повезёт, система автоматически разместит на этой странице условные обозначения входных и выходных соединений в соответствии с расположением контактов на базовой странице и присвоит им определенные названия. После этого можно приступать к созданию принципиальной схемы этого блока (**Рис. 8.13**).

На практике каждый блок может содержать вложенные блоки более низкого уровня или схему на логических вентилях, или (очень часто) их совокупность. Такие иерархические принципиальные схемы позволяют устранить недостатки, присущие одноуровневым схемам, а именно:

1877 г. *Начала работу первая коммерческая телефонная сеть.*

- Облегчают наглядное представление облика устройства и упрощают работу со схемой.
- Упрощают повторное использование блоков в последующих проектах.
- Если часть схемы встречается в конструкции несколько раз, следует реализовать её в виде отдельного блока и при необходимости использовать, т. е. вызывать. Такой подход существенно упрощает работу в случаях, когда в схему блока требуется внести какие-либо изменения. При этом модифицировать придется содержимое только одного блока.

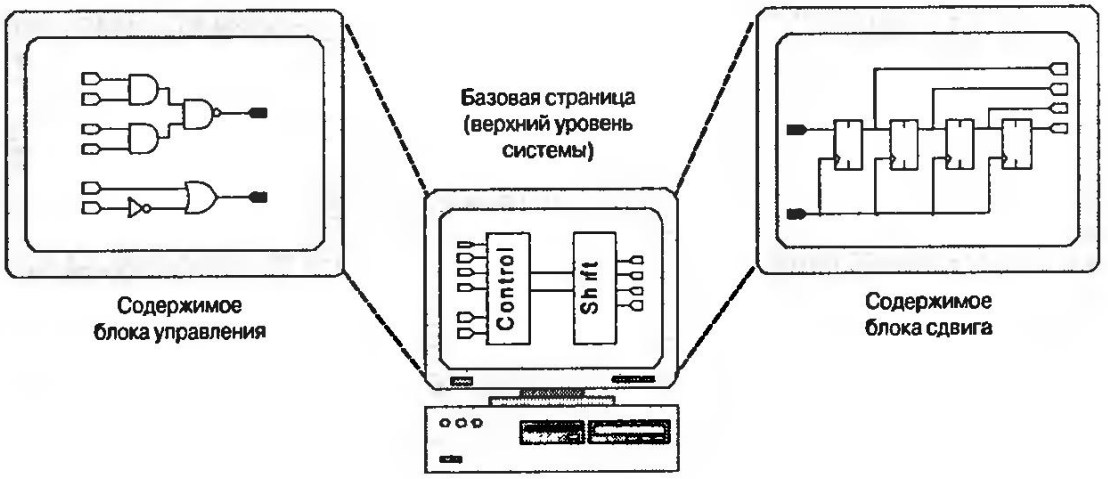


Рис. 8.13. Простая иерархическая принципиальная схема

Современная последовательность схемотехнического проектирования ПЛИС

Раньше каждый поставщик ПЛИС разрабатывал собственное программное обеспечение для ввода и описания принципиальных схем, сопоставления, компоновки, размещения и разводки элементов. Однако, как показала практика, одна и та же компания может преуспеть только в одной сфере — либо в разработке САПР электронных устройств, либо в создании ПЛИС, но не в нескольких областях одновременно.

Другая сторона этой проблемы заключалась в том, что изначально средства проектирования заказных микросхем (ASIC) были слишком дорогими (даже утилиты для ввода принципиальных схем, которые в наши дни расценивают как продукт широкого потребления). В то же время поставщики ПЛИС всегда продавали свои средства разработки по очень низкой цене (достаточно крупным покупателям полный набор компонентов для системы проектирования доставался бесплатно), так как все свои усилия данные компании направляли на продажу микросхем. Конечно, это предложение было очень привлекательно для потребителей, но со стороны поставщиков ПЛИС этот шаг был не очень удачным, так как они делали огромные вложения в системы проектирования, которые приносили слишком маленькую прибыль.

Со временем сторонние создатели САПР электронных устройств стали поставлять программные продукты, начиная со средств описания схемы и кончая утилитами расстановки компонентов, включая и средства логического синтеза (гл. 9 и 19). Производители ПЛИС по-прежнему поставляют менее сложные (если сравнивать с современным уровнем развития) средства проектирования, подобные программам ввода схем, которые входят в состав их базового набора САПР, а также поддерживают средства размещения и разводки элементов.

Однако в наши дни многие инженеры уже давно не начинают процесс создания устройства со схематического описания, выполняемого на уровне логических вентилей. В некоторых случаях поставщики ПЛИС всё же предлагают незначительную поддержку для реализации схематического подхода применительно к современным устройствам, но в основном ограничиваются предоставлением схемотехнических библиотек только для устройств предшествующих поколений. Тем не менее, схематическое описание принципиальных схем всё ещё находит применение у старшего поколения инженеров, а также при необходимости внесения несущественных изменений в действующие устройства. Более того, графические механизмы, которые применялись в ранних программах ввода принципиальных схем, по-прежнему находят применение и на современных этапах проектирования, как будет описано в следующей главе.

1877 г. *Америка. Томас Batcon (Thomas Watson) разработал телефонный звонок для вызова пользователя.*

1878 г. *Америка. Начала работу первая междугородняя телефонная линия между городами Бостон и Провиденс.*