第 1 章

数字电路版图

内容提要

以下是你在本章中将要看到的内容：

- 自动版图设计软件简介
- 为什么自动版图设计只适用于特定单元
- 了解电路是否真如它所应当的那样工作
- 如何能够预知你的平面设计的好坏
- 自动设计程序遇到了问题
- 故障排除提示
- 哪些网络节点应当首先连接
- 哪些网络节点应当手工连接
- 保证版图符合规则的技术
- 数字电路版图设计流程
- 许多反馈环节
- 如何在大的单元中布置电源线
- 布线和时序间的“鸡生蛋”问题
- 你是否真的实现了设计意图
- 如何设计测试用的快捷芯片

引言

现今设计的大多数集成电路都是很大的。我是指 CMOS 数字芯片确实非常大。一个芯片中毫不夸张地可能含有成千上百万个晶体管。要把这样一个芯片用手工进行版图布局已超出了一个掩模设计者的能力范围——至少在任何合理的时间范围内是如此。

因此，大多数大规模的数字芯片都依靠计算机辅助工具来进行版图设计。了解这些数字电路版图的自动设计工具如何工作，能帮助你培养起在数字电路版图设计——甚至是模拟电路版图设计中熟练的技巧习惯。如果已经知道这些软件如何工作，你就可以更快更好地设计电路版图，弥补这些软件的不足，并在障碍出现之前就避开它们。

设计过程

让我们来设计一个数字芯片。在本章中，我们将跟随一个设计小组，利用一套软件工具，从构思一个数字电路开始，通过模拟测试直到最后对该芯片进行实际的逻辑门布置和布线。

让我们从电路设计者的第一步工作开始。

验证电路逻辑

电路设计者一般都采用 VHDL 或 Verilog 语言来设计规模庞大的数字电路。VHDL 代表 VHSIC（Very High Speed Integrated Circuits，超高速集成电路）硬件描述语言（Hardware Description Language），它自 1987 年就已成为 IEEE 的标准。Verilog 是另一种有专利权的逻辑描述语言。在我们的例子中将采用 VHDL 语言。

电路设计者利用 VHDL 语言来构思一个芯片，这一芯片最初只是表现为由数字构成的一个数据库。电路设计者的 VHDL 文本非常类似于 C 语言。[①] 例如一个文本文件的基本意思是，“我需要一个相加两个 16 位数的电路功能。”VHDL 文件就是以这种方式来描述我们的微处理器、数字元件以及需要的任意功能元件。

接下来，这些 VHDL 数据文件被提交给一个计算机模拟器，来测试这一仍为软件形式的芯片电路。VHDL 编码的逻辑功能运行得非常快，要比通常晶体管级的 SPICE 模拟快得多（但还不如真正的硅芯片快）。

VHDL 模拟器要求它所使用的每一个逻辑功能有与工艺有关的软件描述，如上升时间、下降时间、门传播延时等。这些信息以及其他器件参数存放在 VHDL 模拟器可以调用的一系列文档中。除了这些电气描述外，文档中还有每一个门的物理描述，可供模拟器和逻辑综

① 即计算机 C 语言。

合器使用。所有这些文档统称为**标准单元库**或**逻辑库**。

VHDL 编码片段

```
architecture STRUCTURE of TEST is
component and2x
port (A, B, C, D: in std_ulogic := '1';
Y: out std_ulogic);
End component;
Constant VCC: std_ ulogic := '1';
Signal T, Q: std_ulogic_vector(4 downto 0);
begin
T (0) <= VCC;
A1: and2x port map (A => Q (0),B => Q (1),
Y => T (2));
A2: and2x port map (A => Q (0),B => Q (1),
C => Q (2), D => Q (3), Y => T (4));
Count <= Q;
```

供应给你硅片的公司通常也提供一个标准单元库。理论上讲，你得到的单元库是并且将一直是正确无误的。然而，对单元库的更新是经常进行的，单元库的改变可能会使一个曾经正确无误的芯片不再工作，特别是如果在库的更新中有错误的时候。在一个设计项目进行到一半的时候更新库通常是很糟糕的。

通过检查 VHDL 的模拟结果，我们可以在把芯片设计付诸实际生产之前对电路进行调整。这可以大大节省时间和经费。

编译网表

电路设计者一旦完成了对逻辑设计的验证，就可以把 VHDL 编码输入到一个硅**编译器**或**逻辑综合器**。编译器把这一高层次的类似于 C 语言的编码转变为一个文件，它包含所有需要的逻辑功能以及它们应当如何相互连接的信息。

这个文件的基本意思是："为了相加两个 16 位的数，我需要 25 个门，并且它们应当这样相连。"我们所有的逻辑功能就是以这种方式生成并相互参照的。

至此，我们知道了需要哪些逻辑门，还知道了它们最终应当如何相互连接。这一文件称为网表，它将驱动你的自动版图设计工具。

网表片段

```
module test ( in1, in2, out1);
input in1, in2;
output out1;
wire \net1 , \net2 , \net3 ;
AND2_2X U1 ( .Z(net1) , .A(net2) , .B(net3) ) ;
AND4_2X U2 ( .Z(net1) , .A(net2) , .B(net3) ,
.C(net2) , .D(net1) ) ,
endmodule
```

设计者在开始编译 VHDL 编码时将控制各种选择“开关”。这些“开关”控制着如面积、功耗和速度这样一些参数。根据芯片的要求，电路设计者可以决定在编译 VHDL 时只优先考虑速度，或面积，或功耗，或他们所关注参数的某种特定组合。编译结果将因优先考虑问题的不同而不同，所以设计者要在编译开始之前事先选择这些“开关”。

驱动强度

编译器可以生成极大的网络。例如一个具体网络可以有成千上万个单元。一个网络的单元数越多，驱动它们所需要的功率就越大。如果我们试图用单个驱动源来驱动太多的门，那就有可能使驱动晶体管过载，电路将不能工作。

所以在开始生成版图之前，我们先要对网表进行修正，以保证这些大网络的驱动适当。为此，要把驱动网络的单元更换成逻辑功能相同但驱动能力更大的单元。驱动能力是指单元的**驱动强度**或**扇出**。扇出数表示一个门能驱动多少个器件。一个库中的任何单元都可以成为驱动门。

例如，我们可以看到在单元库中有 10 个或 15 个不同尺寸的反相器。这些可供选择的反相器可以分别称为 1x、2x 或 4x 反相器等等。反相器的这些代号表明了它们的驱动强度。由于 1x 反相器通常可驱动两个门，所以 2x 能驱动四个门，而 4x 则可驱动八个门。

你可能会奇怪为什么不用一个大的门来应对电路中所有可能发

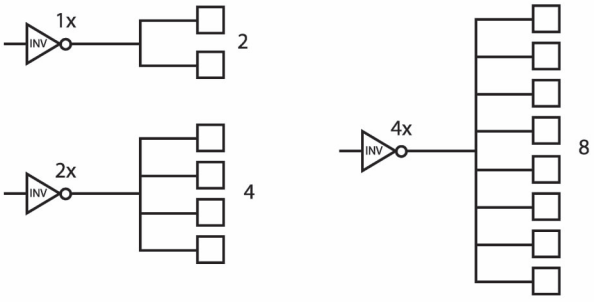

图 1-1　一个反相器驱动两个负载，所以一个 2x 反相器驱动 4 个负载，而一个 4x 反相器驱动 8 个负载。

生的情况。我们是可以这样做的，但这会浪费电路面积并不必要地消耗过多的功率。最聪明的技术应该是确保你所提供的驱动必要而不多余。所以，在编译过程中，编译器会检查每一网络中的门数，并相应地调整驱动每个网络的门的尺寸。如果一个网络太大，用我们驱动强度最大的器件还难以驱动，那么编译器就会把这个网络分割成几个比较容易驱动的较小部分。

缓冲单元

如果编译器把一个大的网络分割成较小易驱动的部分，它将插入额外的门来驱动每一个新形成的小网络。这些额外的门并不是原有逻辑的一部分。电路设计者没有添加它们，你也没有。这是编译器自己做出的决定。

这些额外的门称为**缓冲**单元。缓冲单元帮助驱动门和布线电容，本身却没有什么逻辑功能。无论什么逻辑信号输入到缓冲单元，输出还是同样的逻辑信号。

在下一节，我们将看到编译器如何运用这些概念来驱动一个大的时钟网络的例子。

时钟树的综合

大多数数字电路都以一个嘀哒嘀哒摆动的时钟波形为基准。每一个功能都与这一时钟同步。这一时钟序列信号的布线网络称为**时钟网络**。时钟网络一般都很大，通常它要连接成千上万个门。

不可能建立一个单元使它具有足够的驱动强度来驱动时钟网络中所有的门，所以我们必须再做一些额外的工作使时钟网络能够工

作。正如前面提到的，我们可以把时钟网络分割成较小的部分，再加入缓冲器。这一网络被分割成枝状，称为**时钟树**。建立时钟树的过程称为**时钟树综合**。

为了说明时钟树的概念，让我们来看一个小例子。假设某一时钟网络上有 6 个门，而我们库中所能提供的最大驱动强度即扇出数为 3。因此我们不能指望用一个门来驱动整个时钟网络。所以我们把这个网络分割为两个较小的部分，再分别驱动。

在图 1-2 中，编译器在电路中加入了两个低一级的缓冲器，每一个缓冲器驱动一组三个门。编译器又加入了一个高一级的缓冲器来驱动这两个低一级的缓冲器。所以在我们的电路中已增加了三个额外的单元。

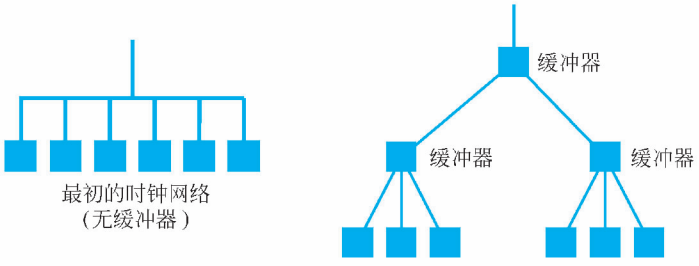

图 1-2 在较小的门组合上加上缓冲器以帮助驱动信号。

如果我们的时钟网络比这还大，编译器就会以这种方式继续进行，分割网络并加入额外的缓冲器，每一个缓冲器驱动其他器件的数目不超过三个。你可以看到，这将形成一个有很多级的很大的树。

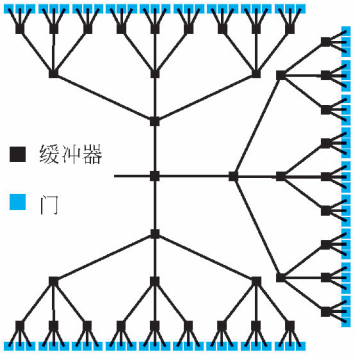

图 1-3 把大的网络分割成许多较小的部分以便于驱动。

随着大量缓冲单元的加入，这些额外的单元将带来原来模拟中未

曾考虑的额外延时。不仅是这个网络，其他大的网络也会要求综合同样类型的树结构，这会增加更多的缓冲器从而也增加了延时。所以一旦综合了时钟网络或者其他大扇出的网络被缓冲时，我们就需要用编译出的网表重新模拟我们的设计。因此编译后有必要再重新模拟。这种反复迭代在芯片开发过程中十分常见。好在这一过程不会永无止境，到某一时刻，你将得到一个最终的网表。

现在一切就绪，我们可以开始版图设计过程了。我们将从平面布局开始。

版图设计过程

现在我们已经可以使用一组或一套称为**布局和布线工具**（**place and route tools**）的自动软件工具。布局和布线工具包含有帮助你完成最终版图的从高层次到低层次的全部软件。正如其名，这些工具主要用来布置门和布置导线，此外还具有一些其他有用的功能。

平面布局（Floorplanning）

我们在布局和布线工具中要用到的第一个软件称为**平面布局工具**。它将帮助你在芯片上划出功能区域、确定这些区域间的连接关系、确定你的 I/O 压焊块的位置，并反馈给你有关平面布局在进行布线时的难易程度。平面布局工具从编译软件产生的网表文件中获取它所需要的连接关系和门的信息。

让我们更细致地跟踪平面布局工具的工作，就从你最初的决定开始。

功能块的布局

通常，你的芯片将被划分成不同的功能区。例如，如果你设计的是一个大的数字芯片，那么在芯片上可能会有微处理器单元（MPU），也许有一个浮点单元（FPU），还可能有一个 RAM 模块和一个 ROM 模块。

每一个功能区放在哪里是由你而不是由计算机来决定的。你也许会说：“好吧，我想把微处理器所有的门放在左下角，把 RAM I 所有的门放在右上角。”等等。以后你还将有可能改变这些决定，那就是在你一旦看到这些决定可能会对版图，特别是布线产生什么影响的时候。

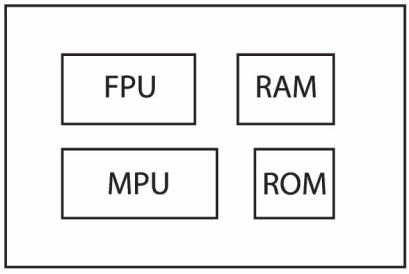

图 1-4 芯片具有非常明确的功能区。

门的分组

一旦确定了功能区，你要做的第一件事就是把每一模块中的门在一定程度上放在一起。例如，你不会希望 FPU 的门散布在 ROM 或 RAM 模块的各处。所有相关的门应当相互间就近放置。

平面布局工具就从帮助你把门放在一起开始。但每个门的确切位置还不在这时确定，我们还不需要这一层的细节。此外，我们也许在以后的某个时候还要改变模块的位置。所以在目前确定大致的相邻位置就可以了。

模块级的连接关系

接着，平面布局工具将帮助你布置芯片的输入和输出(I/O)单元。例如，你也许希望所有通往 FPU 的输入都靠近在左上角的 FPU 模块。为了帮助你做到这一点，有些工具会自动把 I/O 单元切实放在合适的区域；而另一些工具将根据你所确定的布局反馈给你相应的图形信息。

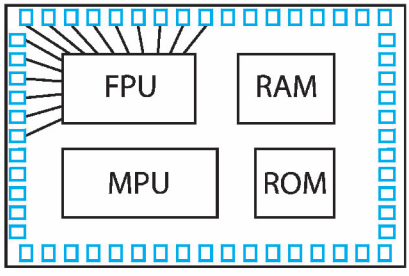

图 1-5 输入和输出可以放置在它们适当的单元模块附近。

平面布局工具还显示了模块之间必须的布线连接。例如它将显

示 FPU 和 RAM 模块之间的连接。

当给以同样的信息时，模拟器和布局布线工具这样的自动软件程序做出的决定，与一个认真负责的设计人员用足够的时间所做出的选择可以是一样的。（我们希望如此。）然而，你会看到经常进行人工干预和监督是非常重要的。

没有人的指导，软件永远也不会工作。即你已经从广义上确定了高层次功能和输入输出的位置。根据芯片特性、最终将被放入的封装尺寸、具体的电路，还有最主要的是对软件工作的理解，你已经为该软件预先确定了一些基本的版图原则。

工具永远也不会完全靠自身来运转。人类的大脑必须监督工具的工作，否则这些工具将变得毫无用处[②]。

使用飞线

通常，平面布局工具会为你显示从每一个模块连至 I/O 压焊块和其他模块的所有导线。这些数不清的导线就是大多数设计工具提到的所谓**“鼠窝”**或**飞线**。

当你在计算机屏幕上按动鼠标选中、拖动和调整你的模块时，你将看到所有这些导线的连接会随着光标同时移动。

当你用鼠标移动模块时，注意这些飞线。如果这些飞线变得交叉杂乱，你就会知道这个电路很难布线。如果没有许多交叉的飞线，那么它的布线将很容易。

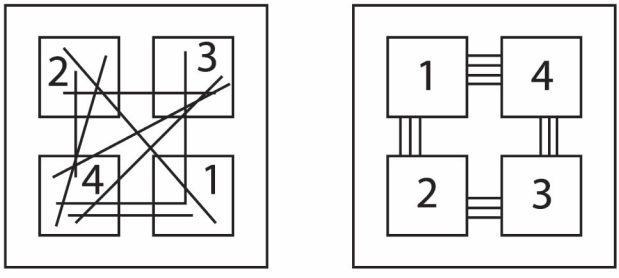

图 1-6　整洁的飞线说明是一个良好的平面布局。

② 正如拼写检查器要结合人的认可一样。

你应该改变模块的平面布局，使飞线看上去尽可能的干净整齐和容易布线。[3] 你也许会决定重新安排整个功能区；也许会把一个小模块移到另一边，并把它安排在两个大模块之间；或者会把位于中间的一个模块放到边上，或是把一个外边的模块放到中间。

当最终得到一个布线简单合适的模块分布图时，你就把平面布局输出文件保存起来。

时序检查

由于最终的平面布局输出文件说明了各个门的总体位置，平面布局工具就会知道所有导线的大致长度。对这些导线长度的估计是根据数字电路库中的实际尺寸做出的。

利用这些信息，平面布局工具可以输出一个包含有导线估计长度的文件，并送回到数字电路模拟器中。现在你可以进行一些模拟，来确定估计的线长对数字电路会有什么影响。你必须检查由于长导线使电路信号减慢太多从而影响电路时序的可能性有多大。

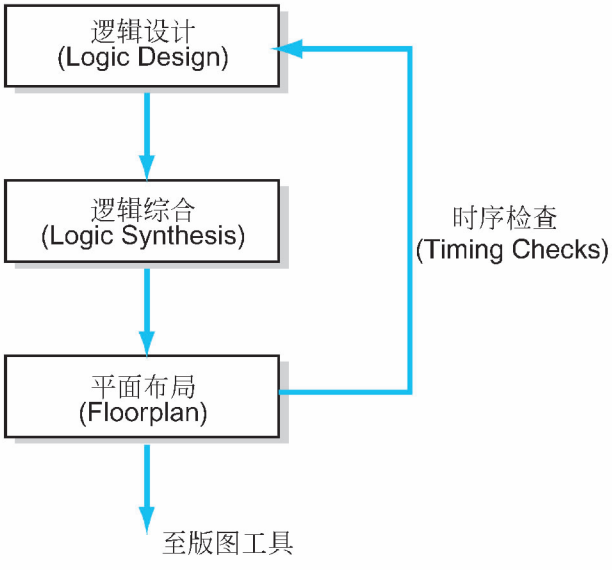

图 1-7 平面布局/时序检查的反复过程。

如果导线长度确实过度影响了电路的时序，那么电路设计者就需

---

③ 正如 Chris 经常说的："我不管它是否行，只要看上去好就可以。"