浙江工业大学

硕士学位论文

基于Cadence平台的ASIC设计与研究

姓名：杨骞

申请学位级别：硕士

专业：通信与信息系统

指导教师：孟利民

20030101

Y 567529

摘 要

本文介绍了在大型 EDA 设计软件 Cadence 平台上设计一种功能较完备较复杂的真空荧光屏显示屏（VFD）驱动芯片。根据“自顶向下”的设计方法，从整个芯片的功能要求出发，将芯片进行了模块划分，结合“自底向上”的原理图输入设计方法对各模块进行具体的电路设计或描述，并对各模块进行功能验证，从而达到设计的目的。

本论文对VFD驱动芯片的关键模块电路：输入接口电路、时序控制电路、命令译码电路、逻辑控制电路、RAM电路、ROM电路的设计作详细的分析，采用了Verilog硬件描述语言，对电路中的逻辑功能与时序关系进行了仿真验证，从而完成芯片的前端设计。芯片版图的设计采用了0.6um硅栅CMOS工艺，设计规则采用首钢NEC的CZ5L工艺规则，高压输出驱动电路部分和输出掩膜电阻部分采用了耐高压DMOS工艺。

最后，该芯片流片成功，并通过了浙江省电子产品检验所的检验测定。

关键词：Cadence　VFD　Verilog-HDL　仿真　版图　工艺

Abstract

Cadence is one of the most famous EDA tools in IC design field. This paper presents the design of one kind of relatively complete relatively complicated vacuum fluorescent display (VFD) controller-driver based on Cadence platform. According to the "top-down" design techniques, beginning from the function of the whole controller-driver, the chip is divided into several modules. With the "bottom-up" graphical schematic techniques, each module is realized by circuit schematic or language description. After the function of each module is verified, the front-end design is now complete.

The key modules including input interface circuit, timing control circuit, commond decoding ciriuit, logic control circuit, RAM circuit, ROM circuit are discussed in detail. And the Verilog hardware description language is used to verify the function of this VFD controller-driver. The SGNEC CZ5L design flow is adopted. CZ5L is SGNEC's optimized triple-metal, double-poly 0.6um CMOS process. And the high-voltage DMOS process is applied by the circuit of high-voltage output and mask resistance.

At last, this VFD controller-driver is tapeoutted successfully, and passes the test of electronic examination and test of products of Zhejiang Province.

Key Words: Cadence VFD Verilog-HDL Simulation Layout Process

第一章　绪论

1．1 集成电路（IC）设计的发展与变革

从 1959 年至今 40 多年来，集成电路技术发生了惊人的变化。它经历了小规模（SSI）、中规模（MSI）、大规模（LSI）、超大规模（VLSI）、特大规模（ULSI）阶段，目前已进入系统集成芯片（SOC）。

作为集成电路产业的基础，这些年来半导体工艺技术一直在得到迅速提高。集成规模按照摩尔定律每 18 个月集成度增长一倍，最小特征尺寸缩小 30%，而芯片尺寸每年提高 12%。表 1.1 反映了近年来的这种发展趋势。其中，0.18μm 的设计技术已在 1999 年提前实现。

年份	1997	1999	2001	2003	2006	2009	2012
最小线宽（um）	0.25	0.18	0.15	0.13	0.10	0.07	0.01
DRAM 容量	256M	1G	1-4G	4G	16G	64G	256G
每片晶体管数（M）	11	21	40	76	200	520	1400
芯片尺寸（平方毫米）	300	440	385	430	520	620	750
频率（MHz）	750	1200	1400	1600	2000	2500	3000
金属化层层数	6	6-7	7	7	7-8	8-9	9
最低供电电压（V）	1.8-2.5	1.5-1.8	1.2-1.5	1.2-1.5	0.9-1.2	0.6-0.9	0.5-0.6
最大晶圆直径（mm）	200	300	300	300	300	450	450

表 1.1　半导体工艺技术的进步及相关特征参数的变化

随着 IC 集成度的不断增加，复杂性日益提高，其设计方法与设计手段也发生了巨大的变革，经历了从原始的全手工设计到目前最先进的计算机全自动实现的整个发展过程。根据各个历史进程中科技进步的特点及各主要设计手段、设计工具的不同，可以把它归为五类，这就是原始的手工设计、计算机辅助设计（CAD）、电子设计自动化（EDA）、电子系统设计自动化（ESDA）和用户现场可编程器件。

1．2 ASIC 设计的方法和流程

ASIC（Application Specific Integrated Circuits）直译为“专用集成电路”。它是面向专门用途的电路，以此区别于标准逻辑（Standard Logic）、通用存储器及通用微处理器等电路。

ASIC 并不是一个学术名词，它的含义很不确切。按字面来解释，凡是用于某一类专用系统的电路都可以称为 ASIC。目前在集成电路界，ASIC 被认为是用户专用集成电路（Customer Specific IC），即它是专门为一个用户而设计和制造的。换言之，它是根据某一用户的特定要求，能以低研制成本、短交货周期的半定制、定制电路以及 PLD 和 FPGA 电路。

1．2．1 芯片设计分类

芯片设计通常分为正向设计与逆向设计两大类。正向设计通常用来实现一个新的设计，而逆向设计是在剖析别人设计的基础上进行某种修改或改进。在这两大类中又可分为“自顶向下”（top-down）和“由底向上”（bottom-up）不同的步骤，如表 1.2 所示：

步骤 / 方法	自顶向下（Top-Down）	由底向上（Bottom –Up）
正向设计	行为设计 结构设计 逻辑设计 电路设计 版图设计	系统划分、分解 单元设计 功能块设计 子系统设计 系统总成
逆向设计	版图解析 电路图提取 功能分析 结构修改 逻辑设计 电路设计 版图设计	版图解析 电路图提取 功能分析 单元设计 功能块设计 子系统设计 系统设计

表 1.2　芯片设计分类表

1．2．2 ASIC 设计的方法

图 1.1 是包括各种设计方法的 ASIC 树。

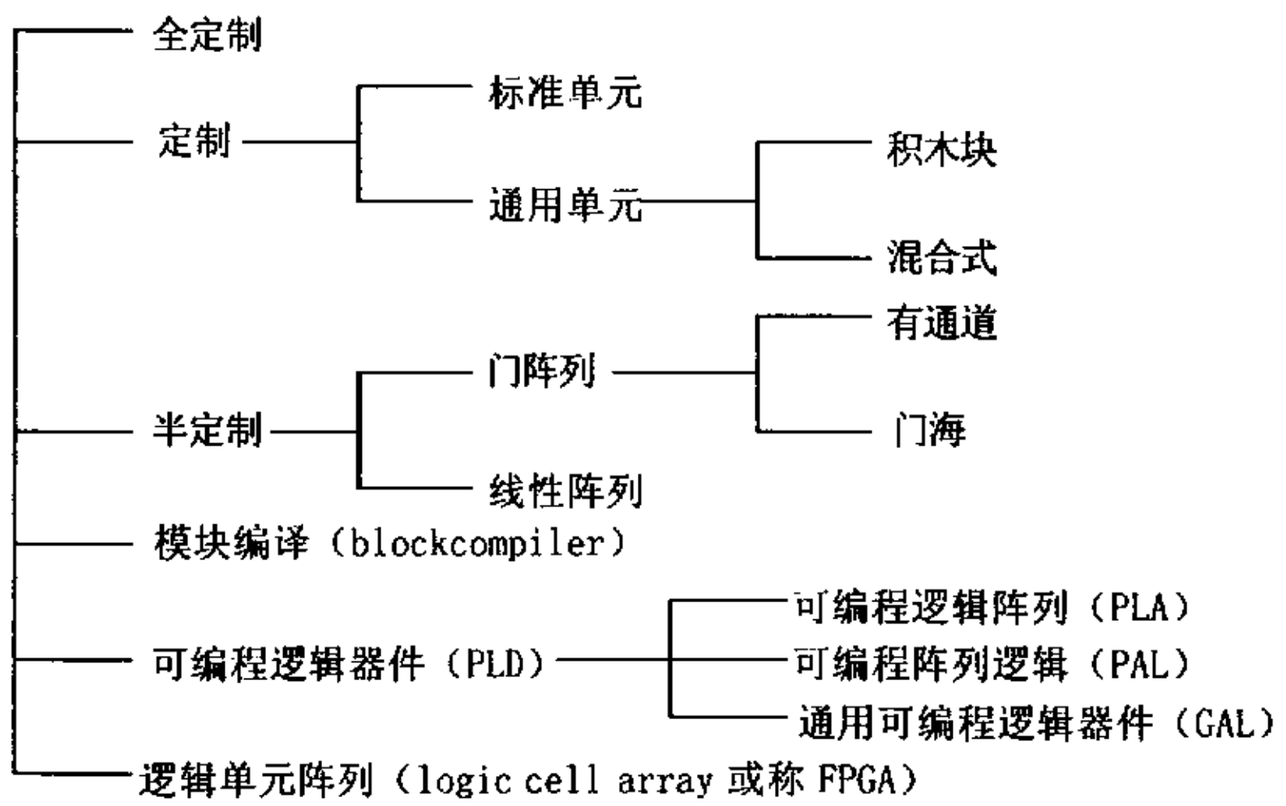


图 1.1　包括各种设计方法的 ASIC 树

1．2．3 ASIC 设计的流程

利用 EDA 工具进行 ASIC 设计是目前较成熟的设计技术，它的主要流程包括以下这些

1．2．1 芯片设计分类

芯片设计通常分为正向设计与逆向设计两大类。正向设计通常用来实现一个新的设计，而逆向设计是在剖析别人设计的基础上进行某种修改或改进。在这两大类中又可分为“自顶向下”（top-down）和“由底向上”（bottom-up）不同的步骤，如表 1.2 所示：

步骤 / 方法	自顶向下（Top-Down）	由底向上（Bottom –Up）
正向设计	行为设计	系统划分、分解
	结构设计	单元设计
	逻辑设计	功能块设计
	电路设计	子系统设计
	版图设计	系统总成
逆向设计	版图解析	版图解析
	电路图提取	电路图提取
	功能分析	功能分析
	结构修改	单元设计
	逻辑设计	功能块设计
	电路设计	子系统设计
	版图设计	系统设计

表 1.2　芯片设计分类表

1．2．2 ASIC 设计的方法

图 1.1 是包括各种设计方法的 ASIC 树。

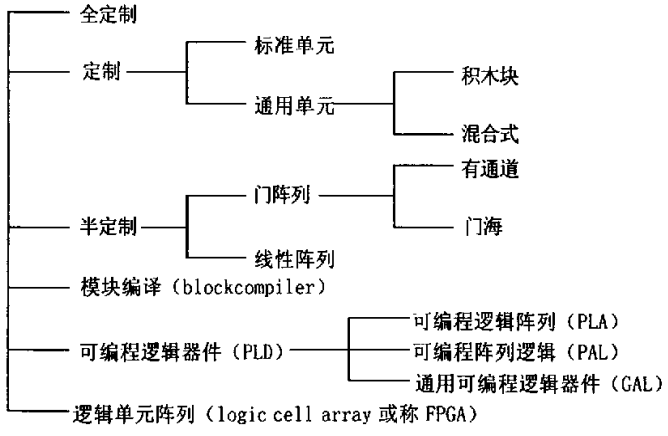


图 1.1　包括各种设计方法的 ASIC 树

1．2．3 ASIC 设计的流程

利用 EDA 工具进行 ASIC 设计是目前较成熟的设计技术，它的主要流程包括以下这些

设计过程：

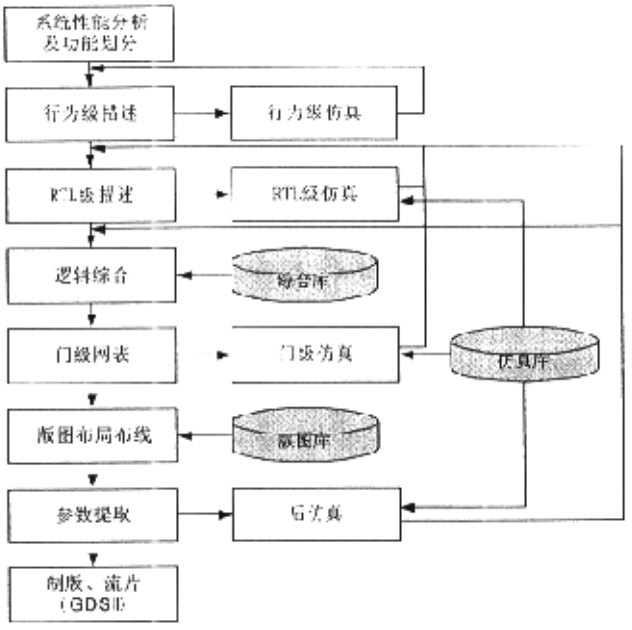


图 1.2　ASIC 设计流程图

1．行为级描述

在完成系统性能分析与功能划分的基础上，对于各个电路功能模块，用 HDL（Verilog HDL/VHDL）完成行为级（Behavior Level）描述。

2．行为级优化与 RTL 级描述的转化

对于上一步中完成的描述进行行为级算法优化与功能仿真，行为级算法优化的目标是选择最优的算法实现方法，行为级仿真的目的是为了验证给定的行为描述是否能够实现所需的功能。在进行行为级优化的同时，通常需要完成向 RTL 级描述的转化。进行这一步转化工作的原因在于，现有 EDA 工具只能接受 RTL 级（RTL：Register Transport Level 寄存器传送级）描述的 HDL 文件进行自动逻辑综合。当然对转化后生成的 RTL 级描述同样需要进行仿真验证。

3．选定工艺库，确定约束条件，完成逻辑综合与逻辑优化

逻辑综合与逻辑优化(Logic Synthesis & Logic Optimization)的目标是将前面得到的RTL级 HDL 代码映射到具体的工艺上加以实现，因而从这一步开始，设计过程与实现工艺相关联。

4．门级仿真

实际上，在 EDA 设计过程的每一个阶段，都需要进行模拟仿真，以期尽早发现并改正错误，保证设计过程的正确性。与前面的行为级仿真和 RTL 级仿真不同的是，完成逻辑综合后的门级仿真包含了门单元的延时信息，因而门级仿真需要相应工艺的仿真库的支持。

5．测试生成

完成逻辑综合后，可产生相应的网表文件（Netlist），但在将设计提交给下一步进行布局布线时，同时应当提供相应的测试文件。测试分为功能测试（Function Test）与制造测试（ManufactureTest）两部分。功能测试就是为了检验线路的逻辑、时序等是否正确。制造测试则是针对半导体工艺而设计的，目的是实现高的故障覆盖率，通常称之为测试向量。

6．布局布线（P&R：Place and Routing）

对于 IC 芯片设计来说，这一步是借助于版图综合的自动布局布线工具，在对应工艺的版图库支持下完成的，通常称之为后端设计。

7．参数提取

在前面完成逻辑综合所产生的门级网表文件中，已经包含了门级单元本身的工艺参数。完成版图综合后，由于布局布线都已确定，可以从版图进一步提取出连线电阻、连线电容等分布参数。

8．后仿真

所谓后仿真，就是将上一步中提取的分布参数再反标到原来的门级网表中，进行包含门延、连线延时的门级仿真。这一步主要是进行时序模拟，考察在增加连线延时后，时序是否仍然满足设计要求。如果不能满足，通常需要回到 3 重新确定约束条件，进行优化迭代。有时侯也可能回到 2，从算法实现上加以调整。

9．制版、流片

在利用 EDA 工具完成设计后，就可交付半导体厂商进行投片生产。

1．3 EDA 工具及发展方向

1．3．1 选用适于 ASIC 设计的 EDA 工具

1．深亚微米技术对 EDA 工具的新要求

尽管 EDA 工具的发展日新月异，但也面临着巨大的技术障碍。这主要体现在高层次综合设计和深亚微米技术方面。

（1）高层次综合设计尚未完善。高层次综合设计中的调度和分配本身就是需要研究的难题。高层次综合设计中整个系统的综合结果常常不能令人满意，如设计空间的有效搜索方法和数字系统的划分等。

（2）深亚微米技术带来的新技术难题。采用深亚微米技术时，由于工艺线由宽变窄，互连电阻和互连电容变大，因此，互连线网络对时序分析的影响和器件延时对时序分析的影响处在同等重要的位置。此外是功耗的挑战，器件在高速运行情况下的开关功耗通常会占到总功耗的 70%～90%，这加重了器件散热的困难，并且导致器件可靠性下降。最后，在精确的延时性能指标下，将无法预期给定面积条件下的可布通性。

2．基于深亚微米技术的 EDA 公司

（1）Cadence 公司是世界上最大的 EDA 工具供应商。Cadence 的后端设计强大，尤其是它的 Drcula 版图验证工具，几乎世界上各大硬件设计厂商都普遍采用。Cadence 新推出的

支持深亚微米设计，基于延时驱动的设计流程及相应的设计工具 Silicon Ensemble，使得 Cadence 在深亚微米的 ASIC 设计领域内更具有优势。

（2）Synopsys 公司的逻辑综合优化工具是目前世界上最强的，它在逻辑综合、行为级综合、测试综合、功耗综合上均有独到优势。

（3）Mentor Graphics 成立于 1981 年，它的前端设计工具比较强，其 QuickSim 2 仿真器速度快，在 FPGA、PCB 及 IC 测试方面有专长。该公司提供了一个称为“商业可重用知识产权库（IP）”，这是一个集成了硬/软件协同设计和强大的系统验证的解决方案，该解决方案能确保芯片和产品在第一次加工时就能正常运行。

（4）Innoveda 公司的前身 Summit 是一家以色列的公司，其前端逻辑设计功能强大，提供可视化的设计输入，能产生被综合的 VHDL/Verilog 代码。Innoveda 的定位是在系统级上，而不是传统意义的芯片级上。

（5）前身是集成电路生产厂家的 Compass 公司，它提供各种通过工艺实践的库模块（如门阵列库、标准单元库以及宏单元库）。

1．3．2 ASIC 设计有关 CAD 技术的发展趋势和难题

虽然 EDA 工具已具有相当高的水平和广泛应用于电子产品设计的每个步骤，但设计技术仍落后于工艺的发展，EDA 工具仍落后于设计的要求。目前的集成电路工艺线条宽度可达 0.25～0.13 um，单片集成可达几百万门。当采用 0.18 um 工艺时，连线延迟已与门延迟相当，以前的布图工具除考虑芯片面积、布通率和布图时间，还要把连线延迟作为重要因素考虑。新推出的设计方法和组装技术如现场可编程门阵列和多芯片模块也提出了新的 CAD 算法和工具的要求。

当前，集成电路 CAD 技术研究呈现的趋势以及存在的难题如下：

1．逻辑综合和高层次综合技术的研究掀起了热潮

当前的发展趋势是把综合和布图、工艺结合起来，这使得问题更难，但也更实用。

2．布图研究向纵深发展

当前的研究主要集中在下列几个方面：一是与时延（Timing Delay）有关的时延约束（Timing Driven）、性能优化（Performance Driven）、时钟偏差（Clock Skew）以及噪声串扰等有关的划分、布图规划、布局、总体布线、斯坦纳（Steiner）树生成及详细布线等算法和布图工具方面的研究及开发；二是多层布线工艺推动的多层布线算法和布线优化的研究；三是由于单个芯片上集成度增加对布图质量和布图效率要求的算法研究，它包括布线均匀性、压缩技术和伪引线端的分配等。

3．传输线的出现给模拟技术带来了变革

在深亚微米工艺下互连线的延迟已超过了门的延迟，在对芯片进行电性能模拟时必须考虑传输线。传输线的延迟模型、关键路径的延迟估算和时延分析是该领域研究的重点。有人利用电磁场理论，引入散射参数来研究传输线模型和计算延迟，并提出了新的模拟算法。

4. 低功耗设计技术冲击各个层次的 CAD 工具

低功耗设计技术要求 EDA 提供各种功耗分析工具以及在各个设计级别进行功率最小化处理（Power Minimization）。功率最小化处理问题包括在电路级减少逻辑门和晶体管级组合电路的开关功耗的优化问题，在逻辑级减少逻辑级组合电路和时序电路的开关功耗的优化问题，在行为级的功耗分析和估算及功率最小化处理问题，在嵌入式系统中从软件的观点减少微处理器的功耗优化问题以及在布图和综合算法中考虑功率最小化处理问题。

5. 新设计方法推动了 DA 的发展

多芯片模块是介于单芯片封装和 PCB 之间的一种封装技术。其中一种最简单的多芯片模块技术是将芯片直接安装在硅圆片上，然后就在硅圆片上制作金属连线以完成芯片之间的互连。它可以克服由于电路复杂性和规模过大而单个芯片容纳不下的困难，但它又比把单个芯片封装在 PCB 上的电性能更好。并且 MCM 易于集成数字和模拟混合的系统，尤其适用于通信系统。

MCM 中的传输线问题和布图问题十分类似于 BBL 布图问题。其主要区别是它更注重于性能驱动的约束要求，包括使芯片延迟最小、特殊线网的布线（如时钟线、电源/地线和母线）以及噪声串扰等问题。

6. 模拟电路的 CAD 工具受到了重视

模拟电路的 CAD 比数字 CAD 更难。近年来模拟电路的硬件描述、综合优化、自动布线等方面都有一定的成果。

7. Fabless 与 EDA 工具

目前从事 IC 设计的公司常采用“Fabless”模式。Fabless 通常指能自己完成芯片设计，交由专业芯片工厂（Foundry）制造，制造好的芯片用自己公司的品牌，并面向客户自行销售。一个“Fabless”类公司的竞争力很大程度上在于其拥有 IP 的多少。“Fabless”模式具有投资小、产值大、核心设计团队只有几个人的特点。

此外，新的研究课题还在不断涌现。CAD 软件工程、设计流程的管理和优化、CAD 的支撑环境、软硬件的协同设计（Software-Hardware Co-design）等都是人们关心的问题。

1．4 本文的主要工作

本文工作的重点是基于 Cadence 设计平台，针对真空荧光显示屏的结构及工作原理，设计真空荧光显示屏显示所需要的专用驱动芯片。

第二章主要介绍了芯片设计所用到的 Cadence 工作平台，以及其在 ASIC 设计中常用的工具。

第三章主要介绍了真空荧光显示屏（Vacuum Fluorescent Display）的结构及工作原理。

第四章详细介绍了 VFD 驱动芯片的电路实现与功能仿真。

第五章详细介绍了 VFD 驱动芯片的版图设计与分析。

第二章 Cadence 工作平台使用介绍

Cadence 公司是全球最大的电子设计自动化公司，它的 EDA 软件在全球计算机、通信、航空航天及民用消费类电子产品的设计、研发部门中获得越来越多的青睐。

Cadence 是一个大型的 EDA 软件，它几乎可以完成电子设计的方方面面，包括 ASIC 设计、FPGA 设计和 PCB 板设计。与众所周知的 EDA 软件 Synopsys 相比，Cadence 的综合工具略为逊色。然而，Cadence 在仿真、电路图设计、自动布局布线、版图设计及验证等方面却有着绝对的优势。Cadence 与 Synopsys 的结合可以说是 EDA 设计领域的黄金搭档。此外，Cadence 公司还开发了自己的编程语言 Skill，并为其编写了编译器。由于 Skill 语言提供编程接口甚至与 C 语言的接口，所以可以以 Cadence 为平台进行扩展，用户还可以开发自己的基于 Cadence 的工具。实际上，整个 Cadence 软件可以理解为一个搭建在 Skill 语言平台上的可执行文件集。

Cadence 包含的工具较多，几乎包括了 EDA 设计的方方面面。这儿介绍一些 ASIC 设计中常用的工具，例如仿真工具 Verilog-XL，布局布线工具 Preview 和 Silicon Ensemble，电路图设计工具 Composer，电路模拟工具 Analog Artist，版图设计工具 Virtuoso Layout Editor，版图验证工具 Dracula 和 Diva，最后介绍一下 Skill 语言的编程。

2．1 Cadence 软件的环境设置和启动

目前 Cadence 软件支持的系统平台只有UNIX系统，我们这儿用的是 SUN 公司的 Solaris 操作系统。要使用 Cadence ，必须在自己的计算机上作一些相应的设置，这些设置包括很多方面，而且不同的工具可能都需要进行各自的设置。Cadence 提供了一个很强大的帮助文件 Openbook ，用户遇到任何问题都可以在 UNIX 提示符下打“openbook &”命令参考其中的 Configuration Guides 及各工具的 userguide 或者 reference ，其访问的方法是 main menu-> System Administration->Configuration Guides 。但作为初学者，只需进行以下几项设置：

1．.cshrc文件：设置 Cadence 软件所在的路径，所使用的 license 文件等。

2．.cdsenv文件：包含了 Cadence 软件的一些初始设置。

3．.cdsinit文件：包含了 Cadence 软件的一些初始设置，一般 Cadence 会自动调用隐含的设置。

4．cds.lib文件：如果用户需要加入自己的库，则可以修改自己的库管理文件cds.lib。

完成了一些必要的设置，就可以启动 Cadence 软件。启动 Cadence 软件的命令有很多，不同的启动命令可以启动不同的工具集，常用的启动命令有 icfb，icca 等。也可以单独启动单个工具，例如启动 Viruoso Layout Editor 可以用 layoutPlus 来启动，Silicon Ensemble 可以用 sedsm 来启动。以 icfb 为例，先在UNIX 提示符下输入 icfb & ，再按回车，经过

第二章 Cadence 工作平台使用介绍

Cadence 公司是全球最大的电子设计自动化公司，它的 EDA 软件在全球计算机、通信、航空航天及民用消费类电子产品的设计、研发部门中获得越来越多的青睐。

Cadence 是一个大型的 EDA 软件，它几乎可以完成电子设计的方方面面，包括 ASIC 设计、FPGA 设计和 PCB 板设计。与众所周知的 EDA 软件 Synopsys 相比，Cadence 的综合工具略为逊色。然而，Cadence 在仿真、电路图设计、自动布局布线、版图设计及验证等方面却有着绝对的优势。Cadence 与 Synopsys 的结合可以说是 EDA 设计领域的黄金搭档。此外，Cadence 公司还开发了自己的编程语言 Skill，并为其编写了编译器。由于 Skill 语言提供编程接口甚至与 C 语言的接口，所以可以以 Cadence 为平台进行扩展，用户还可以开发自己的基于 Cadence 的工具。实际上，整个 Cadence 软件可以理解为一个搭建在 Skill 语言平台上的可执行文件集。

Cadence 包含的工具较多，几乎包括了 EDA 设计的方方面面。这儿介绍一些 ASIC 设计中常用的工具，例如仿真工具 Verilog-XL，布局布线工具 Preview 和 Silicon Ensemble，电路图设计工具 Composer，电路模拟工具 Analog Artist，版图设计工具 Virtuoso Layout Editor，版图验证工具 Dracula 和 Diva，最后介绍一下 Skill 语言的编程。

2．1 Cadence 软件的环境设置和启动

目前 Cadence 软件支持的系统平台只有UNIX系统，我们这儿用的是 SUN 公司的 Solaris 操作系统。要使用 Cadence ，必须在自己的计算机上作一些相应的设置，这些设置包括很多方面，而且不同的工具可能都需要进行各自的设置。Cadence 提供了一个很强大的帮助文件 Openbook ，用户遇到任何问题都可以在 UNIX 提示符下打“openbook &”命令参考其中的 Configuration Guides 及各工具的 userguide 或者 reference ，其访问的方法是 main menu-> System Administration->Configuration Guides 。但作为初学者，只需进行以下几项设置：

1. .cshrc文件：设置 Cadence 软件所在的路径，所使用的 license 文件等。
2. .cdsenv文件：包含了 Cadence 软件的一些初始设置。
3. .cdsinit文件：包含了 Cadence 软件的一些初始设置，一般 Cadence 会自动调用隐含的设置。
4. cds.lib文件：如果用户需要加入自己的库，则可以修改自己的库管理文件cds.lib。

完成了一些必要的设置，就可以启动 Cadence 软件。启动 Cadence 软件的命令有很多，不同的启动命令可以启动不同的工具集，常用的启动命令有 icfb，icca 等。也可以单独启动单个工具，例如启动 Viruoso Layout Editor 可以用 layoutPlus 来启动，Silicon Ensemble 可以用 sedsm 来启动。以 icfb 为例，先在UNIX 提示符下输入 icfb & ，再按回车，经过

一段时间，就会出现如图2.1所示的CIW（Command Interpreter Window）窗口。从CIW 窗口就可以调用许多工具并完成许多任务。

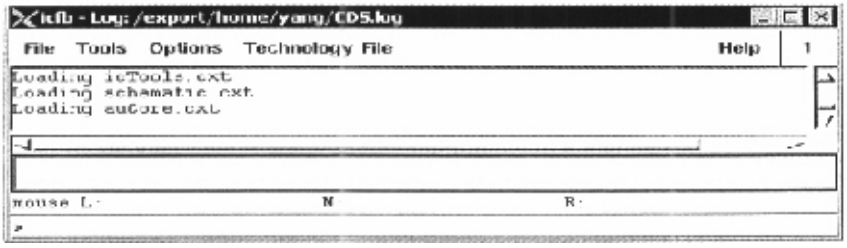


图 2.1 Cadence CIW 窗口

2．2 系统组成

2．2．1 库文件的管理

所有的EDA软件都离不开库的支持。库的丰富程度在一定程度上决定了EDA工具的实用性。如果用户需要加入自己的库，则可以修改自己的库管理文件cds.lib。对于初次使用Cadence的用户，Cadence会在用户的当前目录下生成一个cds.lib文件，用户通过CIW生成一个库时，Cadence会自动将其加入cds.lib文件中。

技术文件库对于IC 设计而言是非常重要的，其中包含了很多设计中所必需的信息。对于电路图、版图设计者而言，技术库就显得更为重要了。要生成技术文件库，必须先编写技术文件。技术文件主要包括层的定义，符号化器件的定义，层、物理以及电学规则和一些针对特定的Cadence工具的规则的定义，例如自动布局布线的一些规则，版图转换成GDSII时所用到的层号的定义。

2．2．2 仿真工具Verilog-XL

人们在进行电子设计时较常用的输入方法有两种，一种为硬件描述语言，一种为电路图输入。随着 ASIC 设计技术的发展，以 HDL 作为输入的设计方法已成为 ASIC 设计的主流。目前较常用的硬件描述语言有 VHDL 和 Verilog 两种。相对而言，Verilog 在工业上用的较为平常。作为 EDA 设计的主流软件之一，Cadence 提供了对 Verilog 及 VHDL 的强大支持。尤其是 Verilog，Cadence 很早就引入了 Verilog，并为其开发了一整套工具。而其中最出色的当数 Verilog 的仿真工具 Verilog-XL。它实现前端功能模拟，用于对 Verilog HDL 描述的语言模块以及数字逻辑电路图进行仿真，它的模拟需要编写一组测试文件来驱动。

2．2．3 电路图设计工具 Composer

一种设计输入的工具，逻辑或者电路设计工程师，物理设计工程师甚至 PCB 板设计工程师都可以用它来支持自己的工作。

2．2．4 布局布线工具 Preview 和 Silicon Ensemble

在自动布局布线前必须进行布局规划（floorplan ），在Cadence 中进行布局规划的工具为Preview ，进行自动布局布线的引擎有四种：Block Ensemble 、Cell Ensemble、 Gate Ensemble 和Silicon Ensemble。其中Block Ensemble 适用于宏单元的自动布局布线，Cell Ensemble 适用于标准单元或标准单元与宏单元相混合的布局布线，Gate Ensemble 适合于门阵列的布局布线，Silicon Ensemble 主要用在标准单元的布局布线中。将Preview 与四种引擎相结合可产生四种不同的自动布局布线环境和流程。

2．2．5 电路模拟工具 Analog Artist(cdsSpice 、Hspice 、spectreS)

实现后端功能模拟，用于对设计好的电路图进行有载模拟，是针对晶体管级的模拟，可进行交流分析、直流分析、瞬态分析和噪声分析。系统提供的模型有cdsSpice、hspiceS、spectreS等等。我们一般用到的是cdsSpice和spectreS。其中采用spectreS进行的模拟更加精确。

2．2．6 版图设计工具 Virtuoso Layout Editor

版图设计的工具，功能强大，可以完成版图编辑的所有任务，版图设计必须考虑具体的工艺实现，因此，放版图的库必须是工艺库或附在别的工艺库上的库。

2．2．7 版图验证工具 Dracula (单独的)/ Diva(在线集成)

版图设计得好坏，其功能是否正确，必须通过验证才能确定。Cadence 中进行版图验证的工具主要有Dracula 和Diva 。两者的主要区别是，Diva 是在线的验证工具，被集成在Design Frame Work II 中，可直接点击版图大师上的菜单来启动。而Dracula 是一个单独的验证工具，可以独立运行。DIVA是Cadence软件中的验证工具集，用它可以找出并纠正设计中的错误：它除了可以处理物理版图和准备好的电气数据，从而进行版图和线路图的对查（LVS）外。还可以在设计的初期就进行版图检查，尽早发现错误并互动地把错误显示出来，有利于及时发现错误所在，易于纠正。DIVA工具集包括以下部分：设计规则检查（iDRC）、版图寄生参数提取（iLPE）、寄生电阻提取（iPRE）、电气规则检查（iERC）、版图与线路图比较程序（iLVS）。

Diva 中寄生元器件提取语句介绍 ：measureParasitic--这个函数通过测量层次或层次之间的关系来获得寄生参数；multiLevelParasitic--这个函数通过测量组合层次图形的面积或边长来生成寄生电容；measureFringe--这个函数一般是在层的边缘按照相应的 DRC 语句来进行测量，他不能测量有覆盖关系的层次，一般用它来描述边缘寄生电容或是侧墙寄生电容 ；calculatParasitic--这个函数可以在前面 measureParasitic 语句所导出的值或是 alculatParasitic 语句所计算出的值的基础上进行进一步的计算 ，它允许我们对更多的简单测量语句进行组合以形成复杂的寄生参数测量；saveParasitic--将测量值作为寄生器件保存到 extracted view

中，在 view 中相应位置会产生相应器件，而这些测量值将作为属性保存；attachParasitic--这个命令将 measureParasitic 命令中测量出的寄生参数值作为属性赋给器件。

2．2．8 Skill 语言程序设计

Skill 语言是Cadence 公司自己开发的一种类似于leap 语言的编程语言。Cadence公司不仅开发了全套的Skill 语言语法，还开发了一个完整的Skill 语言的编译器。整个Cadence 软件都是基于Skill 语言的，所有的Cadence 的工具都是用Skill 语言编写的，甚至各种设置辅助文件都是遵从Skill 语言的语法。此外，Cadence 公司为其每个产品都提供Skill 语言访问函数，这使得用户可以通过Skill 语言访问Cadence 的产品。由于Skill 由于提供编程接口及与C 语言的接口，这使得Cadence可以在原有的基础上进行各种扩展，甚至允许用户开发自己的基于Cadence 平台的工具。

实际上，Cadence 提供了一套完整的编程工具，在CIW 的Tools 菜单下的 Skill Development 可启动该编程环境。

2．3 库的设计

ASIC电路的设计开发过程中需要大量调用库电路单元，所以在设计平台上需要先建立一个新的设计库，然后对所有库单元进行逻辑功能验证，确保正确，在接下来的设计中，就可以直接从库中调用这些单元，在高一层次上对电路进行功能验证，以此类推，直至对整个电路的逻辑模拟。

下面结合 Cadence 工作平台，先为整个设计建立一个新的参考设计库，设计好一些常用的电路单元，并进行功能验证，保证逻辑正确。

2．3．1 单元电路设计

Library (库)的地位相当于文件夹，它用来存放一整个设计的所有数据，像一些子单元（cell）以及子单元（cell）中的多种视图（view）。Cell（单元）可以是一个简单的单元，像一个与非门，也可以是比较复杂的单元（由symbol搭建而成）。View则包含多种类型，常用的有schamatic，symbol，layout，functional，extracted，ivpcell等等。

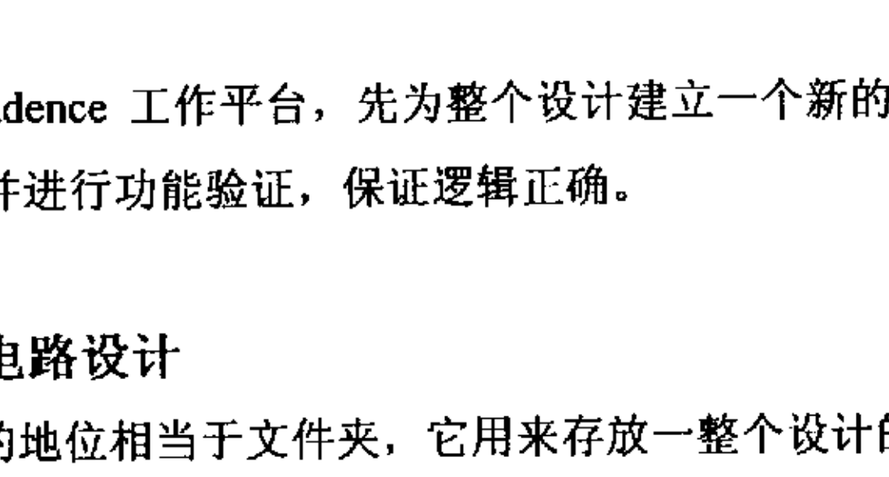


图2.2　库单元结构图

中，在 view 中相应位置会产生相应器件，而这些测量值将作为属性保存；attachParasitic--这个命令将 measureParasitic 命令中测量出的寄生参数值作为属性赋给器件。

2．2．8 Skill 语言程序设计

Skill 语言是Cadence 公司自己开发的一种类似于leap 语言的编程语言。Cadence公司不仅开发了全套的Skill 语言语法，还开发了一个完整的Skill 语言的编译器。整个Cadence 软件都是基于Skill 语言的，所有的Cadence 的工具都是用Skill 语言编写的，甚至各种设置辅助文件都是遵从Skill 语言的语法。此外，Cadence 公司为其每个产品都提供Skill 语言访问函数，这使得用户可以通过Skill 语言访问Cadence 的产品。由于Skill 由于提供编程接口及与C 语言的接口，这使得Cadence可以在原有的基础上进行各种扩展，甚至允许用户开发自己的基于Cadence 平台的工具。

实际上，Cadence 提供了一套完整的编程工具，在CIW 的Tools 菜单下的 Skill Development 可启动该编程环境。

2．3 库的设计

ASIC电路的设计开发过程中需要大量调用库电路单元，所以在设计平台上需要先建立一个新的设计库，然后对所有库单元进行逻辑功能验证，确保正确，在接下来的设计中，就可以直接从库中调用这些单元，在高一层次上对电路进行功能验证，以此类推，直至对整个电路的逻辑模拟。

下面结合 Cadence 工作平台，先为整个设计建立一个新的参考设计库，设计好一些常用的电路单元，并进行功能验证，保证逻辑正确。

2．3．1 单元电路设计

Library (库)的地位相当于文件夹，它用来存放一整个设计的所有数据，像一些子单元（cell）以及子单元（cell）中的多种视图（view）。Cell（单元）可以是一个简单的单元，像一个与非门，也可以是比较复杂的单元（由symbol搭建而成）。View则包含多种类型，常用的有schamatic，symbol，layout，functional，extracted，ivpcell等等。

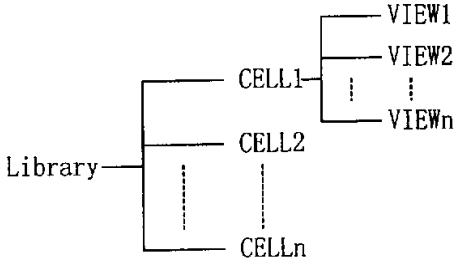


图2.2 库单元结构图

库的建立有两种方法，第一种是通过File菜单，主要的子菜单项有Library、Cellview两项；另一种是通过Tools菜单，主要的子菜单项有Library Manager、Library Path Editor等。

以第一种方法为例：

1．建立库(library)：窗口分 Library 和 Technology File 两部分。Library 部分有 Name 和Directory两项，分别输入要建立的 Library 的名称和路径。如果只建立进行SPICE模拟的线路图，Technology 部分选择 Don't need a techfile 选项。如果在库中要创立掩模版或其它的物理数据（即要建立除了schematic外的一些view），则须选择 Compile a new techfile (建立新的techfile) 或 Attach to an existing techfile (使用原有的techfile)。新建的库是一个空的库，里面什么也没有，用户可在库中生成自己所需的单元。

2．建立单元文件(cell)：在 Library Name 中选择存放新文件的库，在 Cell Name 中输入名称，然后在 Tool 选项中选择 Composer-Schematic 工具(进行SPICE模拟)，在 View Name 中就会自动填上相应的 View Name—schematic。当然在 Tool 工具中还有很多别的工具，常用的象 Composer—symbol、virtuoso—layout 等，分别建立的是 symbol、layout 的视图（view）。在 Library path file 中，是系统自建的 library path file 文件的路径及名称(保存相关库的名称及路径)。

然后就需要建一些最常用的单元，比如，PMOS管、NMOS管、反相器、传输门、触发器、锁存器等等。

例如可以生成一个反相器单元，并为其生成一个电路及版图视图，其流程如下：

1．选择File菜单中的New项，并选择Cellview项，在弹出的对话框中输入单元名inv，并选择视图类型Schematic，再点击ok按钮。

2．用Add菜单中的Component命令调用analogLib中的单元，输入PMOS和NMOS管以及电源和地。如图2.3所示。

3．点击Check and save命令保存。

用同样的流程可生成inv的版图视图。利用Tools中的library manager可以对库进行管理。

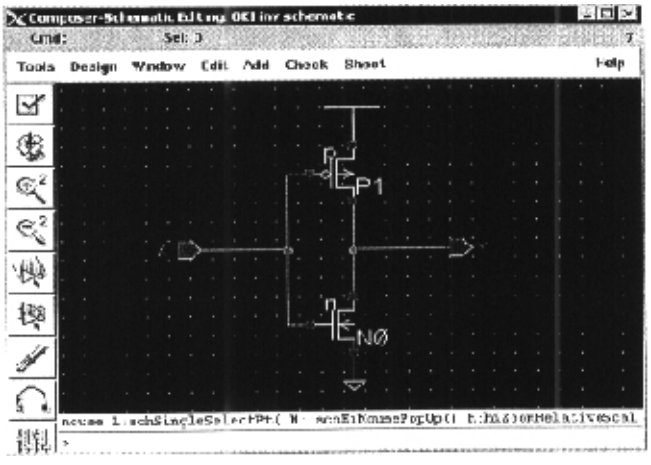


图 2.3　反相器单元电路图

2．3．2 单元符号块的建立

在电路越来越复杂的情况下，如果再花时间去建立一个复杂的 schematic，明显会使工作更繁复。因此我们在建立了一个子电路后，可以将其看作一个整体，建立一个模块，即建立一个 symbol（view name），放在用户自己的库里作为一个器件(component)来用。

下面还是通过子模块反相器的建立，来说明这一内容。

在 Library Manager 中新建 cell ，在弹出的对话框中输入单元名 inv，并选择视图类型 Composer-symbol ，即建立的是 symbol(view)；然后用 Composer 里的一些画图工具画出反相器的基本形状，并添加标签[@instanceName]，如图 2.4 所示，图中的 PIN 的名称必须和上图中的 PIN 名称一致，这样才能建立起一一对应的关系。

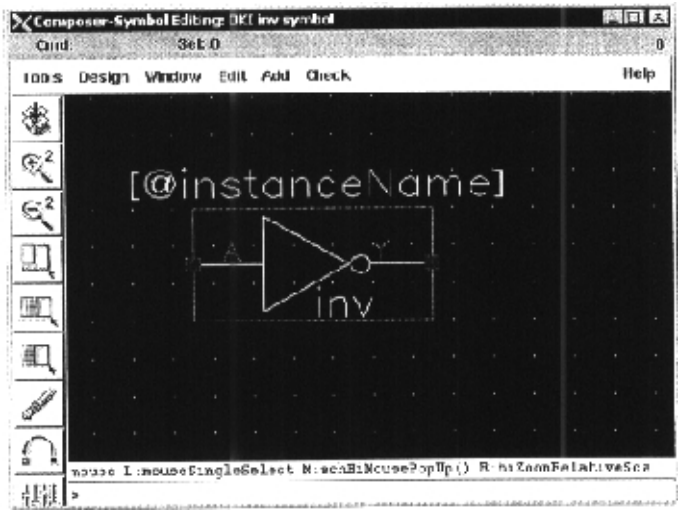


图 2.4 反相器单元符号

在模拟过程中，就可以通过添加元器件（component）来直接将反相器加到电路中来，而不用具体画出其内部的结构，这实际上就是以一个简单的 symbol 来代替其内部的复杂结构。以此类推，可以将小模块一步步的拼凑成大的模块，直接用于模拟仿真。在 Composer 中，symbol 和内部 schematic 的切换是通过 “E” 键来实现的，这样我们就可以很方便的查看每个 symbol 内部的具体电路结构。有一点要注意的是：对于有源器件（如反相器）建立 symbol，必须在原始电路图上添加 analoglib 中的源和地，而且源的电压值也需要设定好，否则变为 symbol 搭成电路后会出错。当然用于模拟时设定的激励源是不用加在电路图中的。

2．4 基于 Verilog-XL 的单元逻辑功能验证

所有库单元都需经过逻辑验证才能在以后的设计中被调用，这一验证过程我们是通过仿真工具 Verilog-XL 来完成的。图 2.5 表示的是仿真过程方框图。

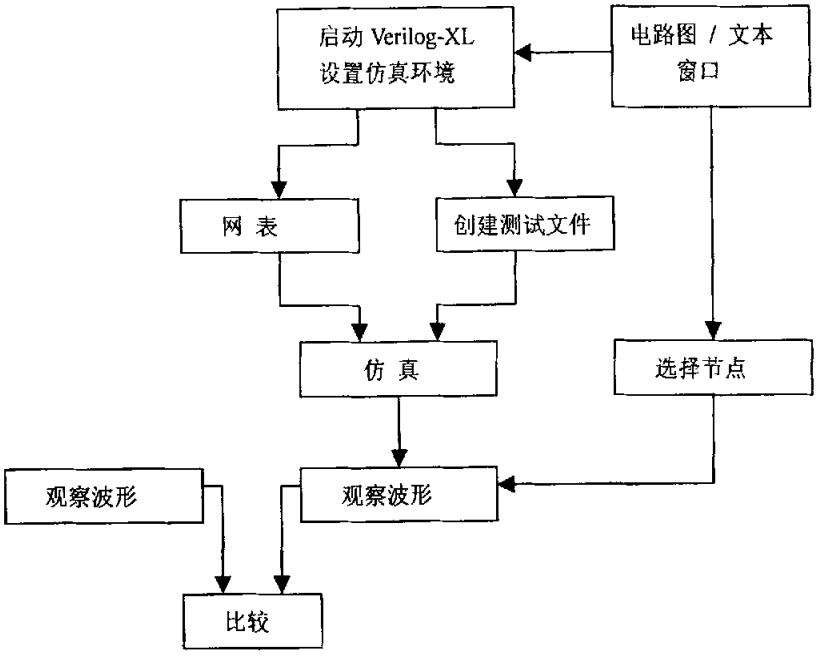


图 2.5 Verilog-XL 仿真流程方框图

启动 Verilog-XL 也有两种方法，一种是从 Composer 窗口中启动，它主要是针对电路图仿真；另一种是在 UNIX 命令提示符下打命令启动，它主要是针对 Verilog 语言描述的文件进行功能仿真，两种方法的显示界面各不相同。

图 2.6 是一个带清“0”端 R 和置“1”端 S 的 D 触发器，它由传输门，或门，三态或门组成。

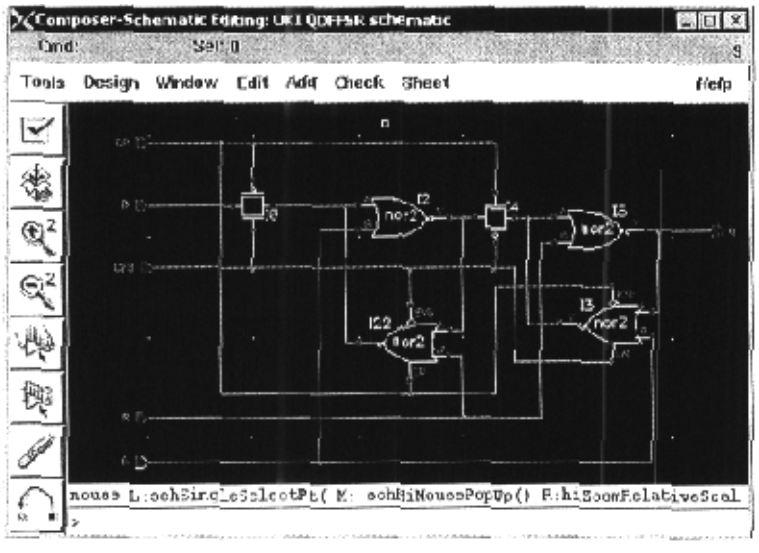


图 2.6　带置位、复位端的 D 触发器单元电路图

这里的传输门，或门，三态或门都是由相应功能电路图生成的模块符号，并且经过逻辑

验证后，从库中直接调用的。它是个上升沿触发的 D 触发器，它的工作原理如下：任何情况下，当复位信号 R 为高电平时，则输出 Q 为“0”，D 触发器被复位；而当 R 为低电平，置位信号 S 为高电平时，由第二级 RS 触发器可以知道，D 触发器被置为“1”，显然，复位比置位的优先级要高；若 R、S 均为低电平，时钟信号 CP 也为低电平时，第一级传输门打开，第二级传输门关闭，则输出信号通过 I3 的反馈，避免 MOS 管栅极上的电荷流失，保持输出状态，直至下一个上跳沿的到来，当 CP 产生上跳沿时，第一级传输门关闭，第二级传输门导通，这时， I2 中的信号就进入 I5 中，并输出，I22 作为一个反馈回路此时有效，目的是增强回路中 MOS 管栅电极上的电荷，避免电荷流失，增强输出信号的强度。

2．4．1 Verilog-XL 下的部分文本描述仿真

Verilog-XL 是基于事件的仿真工具，它只是对电路逻辑做仿真，而对于电荷效应却是没有作用的，而 D 触发器的工作原理涉及到了电荷存储效应，那这个问题该如何解决呢？

这其实在仿真过程中是一个很重要却很容易被忽视的问题，我一开始做仿真的时候就遇到过个问题，得出的结果莫名其妙，事实上就是忽略了这个问题。

解决这个问题的方法有两种，一就是对涉及电荷存储效应的单元写一个 function，它的功能应该等价于该逻辑单元的功能，因为在库管理工具的 View Name 中除了 schematic、layout、extracted 等外，有一种是 function，它是用文本来描述的。

Verilog-XL 可以导入 Functional、Schematic、Netlist、Symbol 四种文件格式，即可以单独对一种格式做仿真，又可以对几种不同的格式混合做仿真，这是因为无论哪种情况，Verilog-XL 最终都会把它转化成它可读的网表文件。

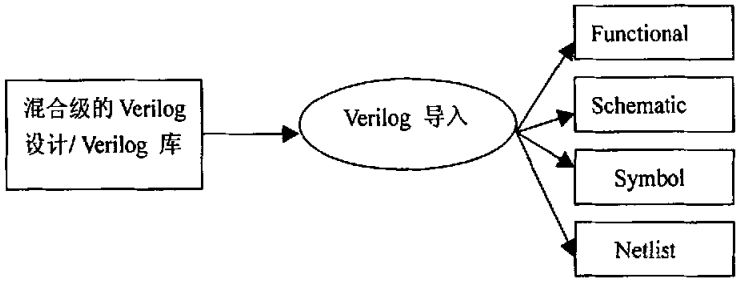


图 2.7　Verilog-XL 的仿真对象格式

Verilog-XL 在对 View Name 中的文件做仿真时，function 的优先级比 schematic 的要高，如果在一个 View 里同时有 function 和 schematic，它就会对 function 做仿真。在这儿就是单独对三态或门写一个 function，它应完全和电路功能一样，仿真时 Verilog-XL 就会自动调用三态或门中的 function 。语言描述如下所示：

```
module trinor2 （ Y , A , B , EN , ENB ）
    output  Y ；
```

```
    input    A ;
    input    B ;
    input    EN ;
    input    ENB ;
    nor   I0 ( Q , A , B );
    cmos   I1 (Y , Q , EN , ENB );
endmodule
```

当电路或 function 都检查无误后，就可以对此 D 触发器的电路进行逻辑仿真了，这里我选择在 Composer 环境中调用 Verilog-XL。首先选择 check and save 按钮，对电路自动做电气检查，防止出现交叉线、短路、断路等情况，然后启动 Tools－Simulation－Verilog-XL 项，这时系统会在该设计项目的当前目录下，自动生成一个仿真文件夹，所有仿真生成的网表文件，测试激励文件，以及日志文件等都保存在此目录下面。仿真器已对电路或 function 产生了一个 netlist 文件， Verilog 的仿真需要用户编写 Testbeach ，此时在“Stimulus－Verilog”菜单中已经产生了测试模板文件 testfixture.verilog ，如图 2.8 所示：

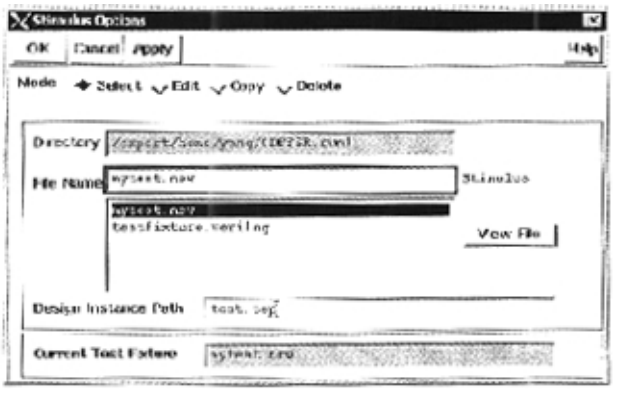


图 2.8　仿真测试文件选择界面

如果我们需要对 D 触发器加不同的激励信号，就需要在 testfixture.verilog 里用语言实现，但这里要说明的是，如果我们直接修改 testfixture.verilog 文件的话，那么每次重新更新网表后，该文件里的激励信号都会回复到初始信息状态，所以我们一般新建一个激励文件，这样，仿真时就不必每次都修改激励文件了，在图 2.8 中，我们新建的文件为 mytest.new 。

testfixture.verilog 文件的初始信息如下：

```
// Verilog stimulus file.
// Please do not create a module in this file.
// Default verilog stimulus.
initial
begin
```

```
    CP = 1'b0 ;
    CPB = 1'b0 ;
    D = 1'b0 ;
    R = 1'b0 ;
    S = 1'b0 ;
End
```

我们对 D 触发器编写的激励文件如下：

```
// Verilog stimulus file.
// Please do not create a module in this file.
// Default verilog stimulus.
initial
begin
    CP = 1'b0 ;
    CPB = 1'b1 ;
    D = 1'b0 ;
    R = 1'b0 ;
    S = 1'b0 ;
end
always #100 CP = ~CP ;
always #100 CPB = ~CPB ;
always #2000 D = ~D ;
always #300 R = ~R ;
always #400 S = ~S ;
```

然后，在 Setup—Record Signals 里面选择 All Signals ，这样我们除了可以看见顶层模块的信号波形，还可以进入各子模块内部，观察内部信号波形。接下来就可以运行 Simulation —Start Interactive ，对仿真文件及激励文件进行编译，结果如图 2.9 所示：

图 2.9　Verilog-XL 综合控制环境

接着我们就可以在 DEBUG 菜单或窗口左边工具栏里设置一些仿真调试选项，“RUN”以后，Verilog-XL 就开始仿真了。仿真结果如图 2.10 所示：

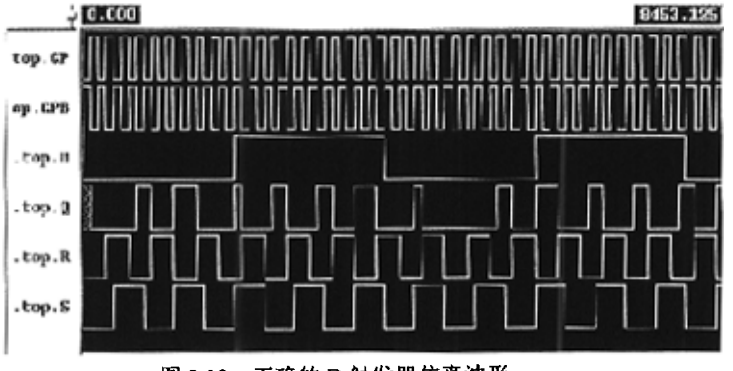


图 2.10　正确的 D 触发器仿真波形

从图中可以看出，仿真波形与电路实际功能完全吻合。

如果没有注意到电荷存储效应这个问题，而用 Verilog-XL 直接对电路进行仿真，则仿真结果如图 2.11 所示：

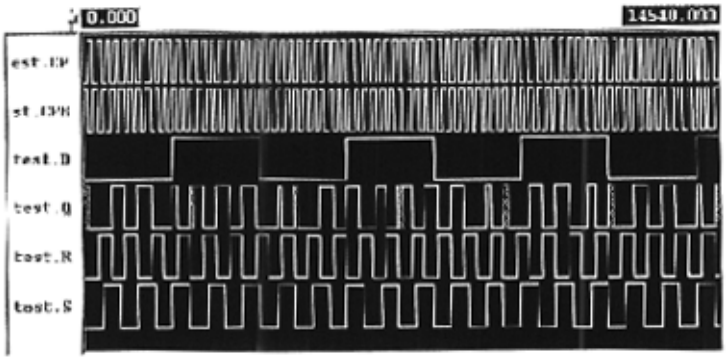


图 2.11　忽略电荷存储效应时的仿真波形

从图中可以看出，当 R 和 S 都出现下降沿时，输出 Q 就会出现不定态，这是由于没有考虑到电荷存储效应而引起的。

2．4．2 Verilog-XL 下的全文本描述仿真

第二种方法就是对整个 D 触发器用 function 来描述，替换掉电路，源程序如下所示：

“dffsr.v”

```
module QDFFSR ( Q , CP , CPB , D , R , S );
    output  Q ;
    input   CP ;
    input   CPB ;
    input   D ;
    input   R ;
    input   S ;
reg  y ;
buf  ( Q , y ) ;
always @( posedge CP or posedge R or posedge S )
 if  ( R= =1 )
    y = 1'b0 ;
else
  if   (S= =1 )
    y = 1'b1 ;
  else
   y = D ;
endmodule
module test;
reg tCP，tCPB，tD，tS，tR;
```

```
wire tQ;
QDFFSR tQDFFSR ( tQ, tCP, tCPB, tD, tR, tS );
initial
$monitor ( $time, "Q = %d CP = %d CPB = %d D = %d R = %d S = %d", tQ, tCP, tCPB,
        tD, tR, tS );
initial
begin
    tCP = 1'b0 ;
    tCPB = 1'b1 ;
    tD = 1'b0 ;
    tR = 1'b0 ;
    tS = 1'b0 ;
end
always #100 tCP = ~tCP ;
always #100 tCPB = ~tCPB ;
always #2000 tD = ~tD ;
always #300 tR = ~tR ;
always #400 tS = ~tS ;
endmodule
```

在这个程序中，把主程序和测试文件写在一起，可以用第二种方法对该语言描述文件进行编译仿真。

下面结合 SimVision 图形环境对此程序作仿真。

1．命令的选择

Verilog 命令有很多参数，在这里，我介绍几个最常用的：

1）-c

首先应该保证所编程序的语法正确性。先进行语法的检查，选择参数- c 键入如下命令：

verilog –c *.v↙

根据 Cadence 的报告，查找错误信息的性质和位置，然后进入文本编辑器进行修改，再编译，这是个反复的过程，直到没有语法错误为止。

2）-s

进入交互式的环境，人机交互运行和下面的参数联合使用。

3）+gui &

verilog 仿真有命令和图形界面两种方式。图形界面友好和 windows 使用很象，很好掌握，一般都使用图形方式。 “&”符号是后台操作的意思，不影响前台工作。如此时

可以在命令行输入其它的命令。

所以这里需要用到的命令就是：

verilog -s dffsr.v +gui &↙　　(*代表文件名)

进入图形交互界面。

2．SimVision 图形环境

SimVision 是 Verilog-XL 的图形环境。主要有 SimControl、Navigator、Signal Flow Browswer、Wactch Objects Window 、SimWave 等窗口。

1）SimControl 窗口

此窗口是主要的仿真控制窗口，让用户和机器进行交互式操作。执行各种 Verilog-XL 命令(菜单)，进行仿真、分析、调试你的设计。该窗口可以显示设计的模块，显示和设置断点、强制信号等。创建用户自己的按钮和执行经常使用的操作。

这儿主要介绍一下 Tool Bar ，图示如下：

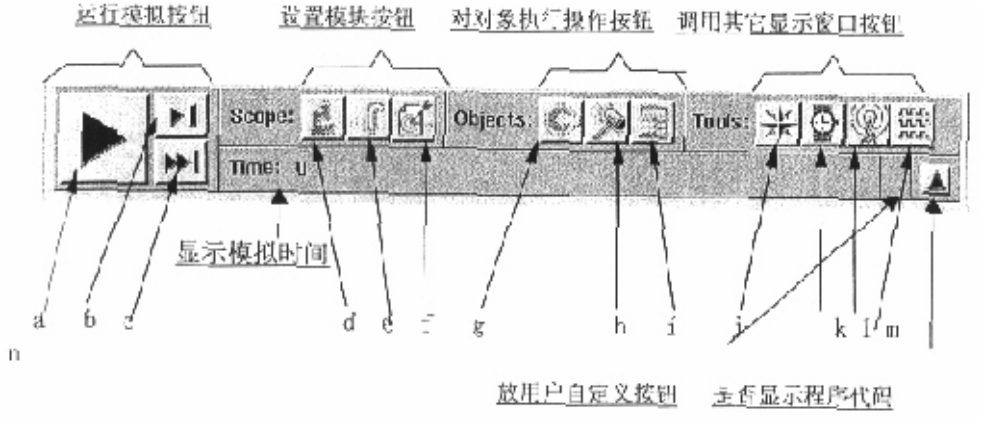


图 2.11　SimControl 窗口中的工具条

a、Run Simulation 按钮

运行模拟，若无断点直至完成，图标变为停止模拟图标。若有断点则运行到断点对应信号再改变的位置。

b、Single Step 按钮

在任何模块每按一下执行到下一个可执行行，即使在子程序中也是单步运行。

c、Step Over 按钮

在当前的模块中执行到下一个可执行行，在子程序中不是单步执行，而是一步执行完子程序。

d、Set Scope 按钮

由当前的调试模块转到被选中的模块。

e、Scope Up 按钮

由当前模块转到它的上一级模块，但若有对象被选中，不执行。

f、Show Execution 按钮

模拟时更新当前模块，显示正在模拟的模块。在当前刚执行完的代码行左边有一个箭头

g、Set Breakpoint 按钮

设置断点，当模拟过程中被选信号变化时发生。代码左边的行号为高亮的可设为断点，灰色则不可以。

h、Set Force 按钮

弹出一个窗口，里面有当前选中信号的名字和数值。用户可以强制信号为一个希望值。

i、Show Value 按钮。

以下 j、k、l、m 调用其它调试窗口，具体介绍放到后面。

j、打开 Navigator 窗口。

k、打开 Watch Objects 窗口。显示被选中的对象

l、打开 Singal Flow Browser 窗口。把被选中的对象放到浏览器中

m、打开 SimWave 窗口。显示被选中对象的模拟波形。

n、程序代码是否显示的切换按钮。显示当前被选信号的数值。

2）Navigator 窗口

此窗口用图形，在 Scope Tree 中采用树的形式显示设计中各模块的层次关系。在 Objects List 中显出 Scope Tree 中被选模块的当前模拟数值和描述。

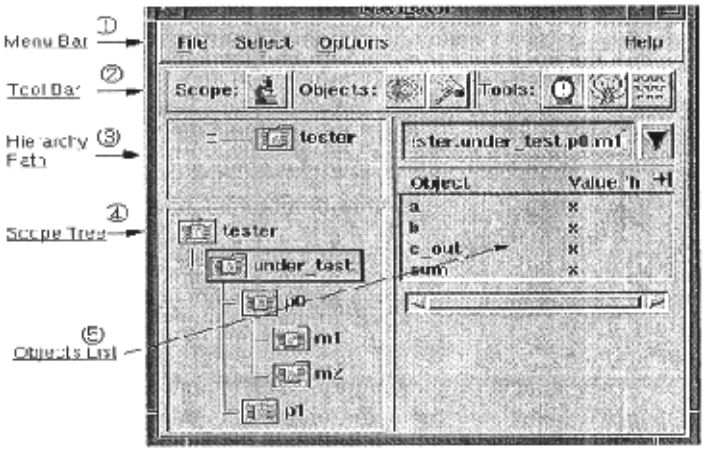


图 2.12 Navigator 窗口

3）Watch Objects 窗口

显示所选信号及其数值，当模拟中断时，更新数值。它可以分别从 SimControl 窗口、Navigator 窗口、Singal Flow Browser 窗口中打开。

窗口菜单项的含义都比较明了，提一下 Options-Heighlight Activity 项使最新变化的信号项用高亮条表示，Options-Continous Update 项使信号随时变化，即使按图 2.11 中的 a、Run Simulation 按钮也会显示最后的结果，否则不显示最后结果。

4）Singal Flow Browser 窗口

该窗口跟踪可疑信号的值，它可以分别从 SimControl 窗口、Navigator 窗口、Watch Objects Window 窗口中打开。

现在就用 SimVision 对 Verilog 描述的 D 触发器作逻辑验证。

1）在命令行中敲 textedit dffsr.v↙　用 textedit 编好程序的文本。

2）在命令行中敲 verilog –c dffsr.v↙　　编译通过程序。

3）在命令行中敲 verilog –s dffsr.v　+gui&↙　进入交互式图形界面 SimControl 窗口。

在 Subscope 中选择 tQDFFSR ，在 Scope 中选择 test.tQDFFSR 。

4）在 SimControl 窗口中选中 Select-Ports 项，选择端口。

5）按下图 2.11SimControl 窗口中的工具条中的 k 键，打开 Watch Objects 窗口，并选中 Options-Continuous Update ，Highlight Activity 两项。

6）按下图 2.11SimControl 窗口中的工具条中的 m 键，打开 SimWave 窗口。

7）按下图 2.11SimControl 窗口中的工具条中的 a 键，得到图 2.14 所示的仿真波形。

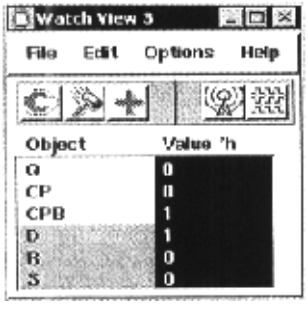


图 2.13　Watch Objects 窗口

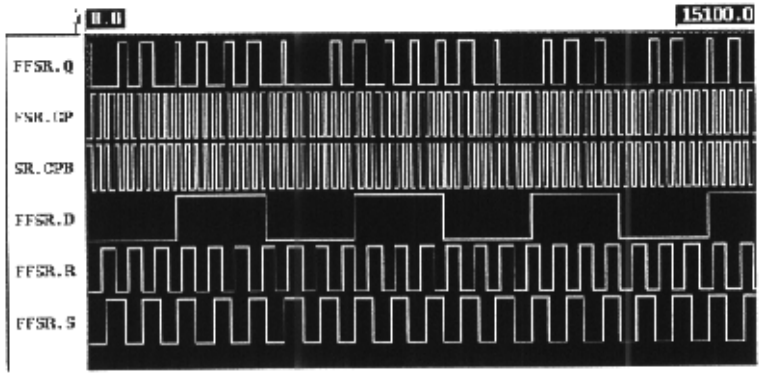


图 2.14　SimWave 窗口波形

由以上的仿真波形图可见，两种方法得出的结果是一致的。

2．5 本章小结

本章主要介绍了芯片设计所用到的 Cadence 工作平台，以及其在 ASIC 设计中常用的工具，例如仿真工具 Verilog-XL，布局布线工具 Preview 和 Silicon Ensemble，电路图设计工具 Composer，电路模拟工具 Analog Artist，版图设计工具 Virtuoso Layout Editor，版图验证工具 Dracula 和 Diva 等。

在本章中结合实际设计的经验，着重介绍了参考库的设计方法、基于 Verilog-XL 的单元逻辑功能验证流程、比较了两种不同的仿真方法，并提出了仿真过程中应特别注意的问题。

第三章　真空荧光显示屏（VFD）工作原理

真空荧光显示屏（Vacuum Fluorescent Display）是从真空电子管发展而来的显示器件，由发射电子的阴极（直热式，统称灯丝）、加速控制电子流的栅极、玻璃基板上印上电极和荧光粉的阳极及栅网和玻盖构成。它利用电子撞击荧光粉，使荧光粉发光，是一种自身发光显示器件。由于它可以做多色彩显示，亮度高，又可以用低电压来驱动，易与集成电路配套，所以被广泛应用在家用电器、办公自动化设备、工业仪器仪表及汽车等各种领域中。VFD根据显示内容可分为：数字显示、字符显示、图案显示、点阵显示。

3．1 VFD 的结构及工作原理

VFD 种类繁多，以其中最被广泛应用的 3 极管构造为例说明其基本构造与原理。

图 3.1 是 VFD 结构的分解斜视图，图 3.2 为剖面图，其构造以玻盖和基板形成一真空容器，在真空容器内以阴极 CATHODE（灯丝 FILAMENT）、栅极 GRID 及阳极 ANODE 为基本电极，还有一些其它的零件（如消气剂等）。

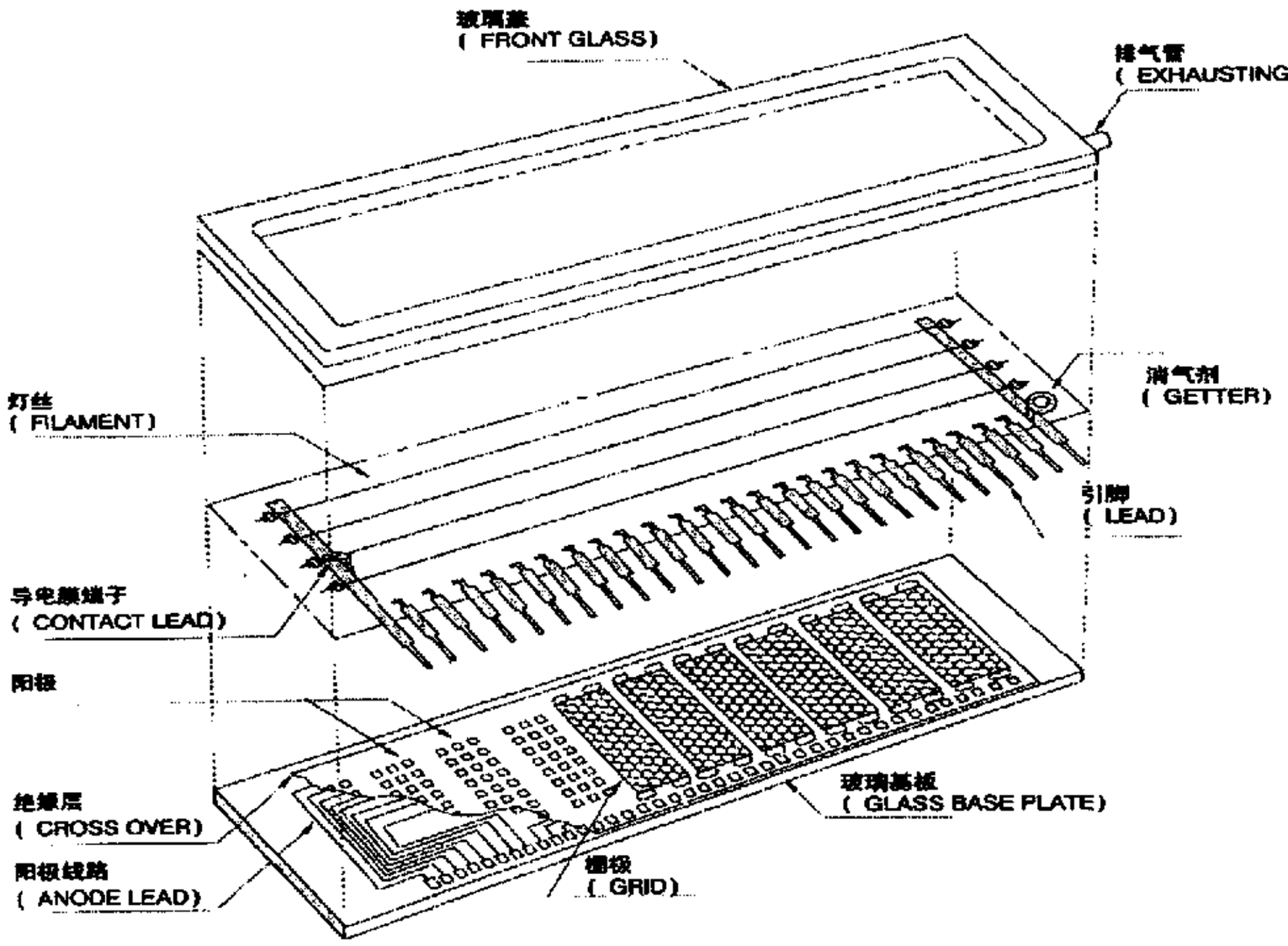


图 3.1　VFD 的分解斜视图

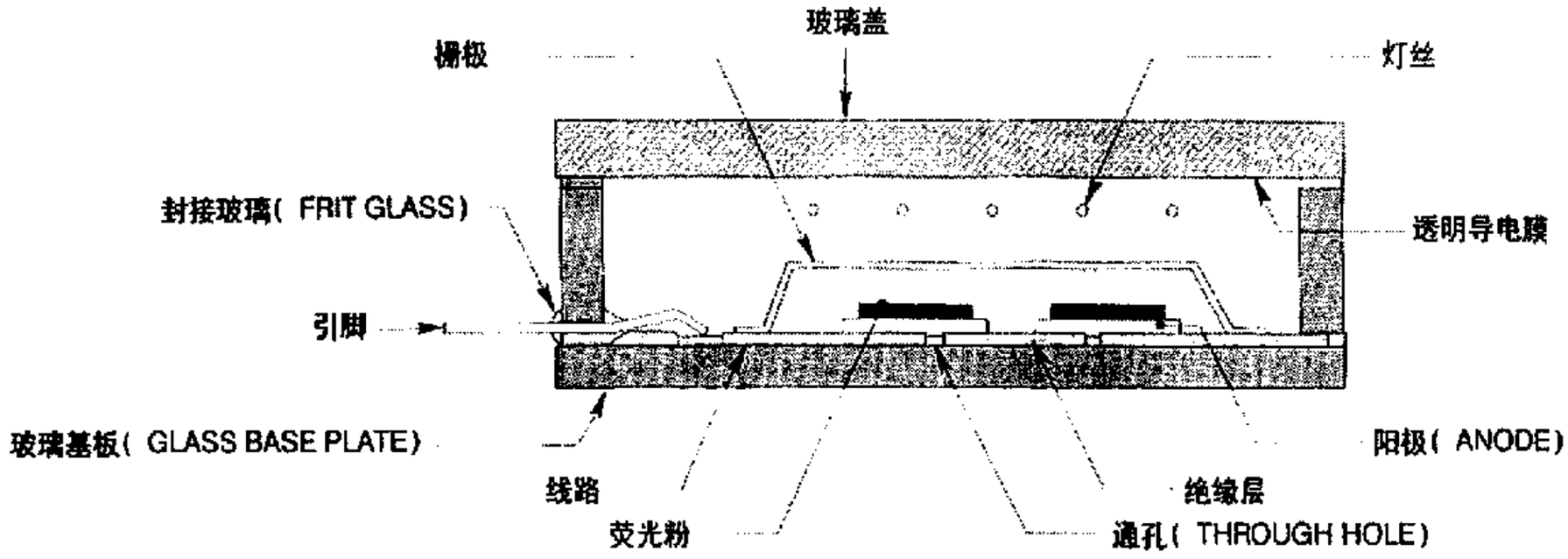


图 3.2　VFD 的剖面图

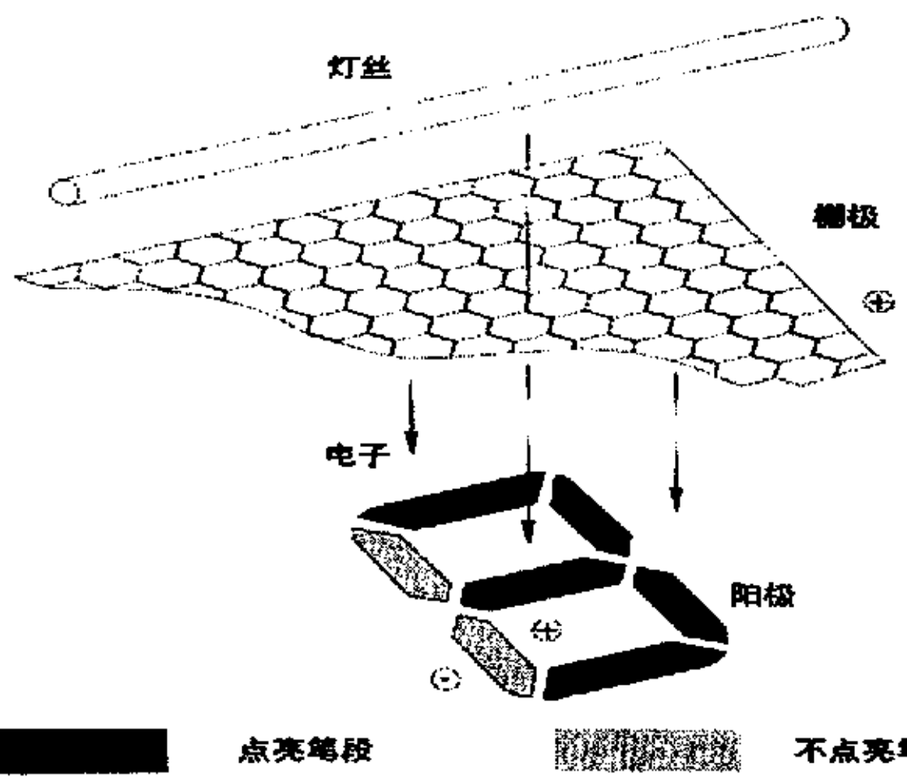


图 3.3　VFD 的基本工作原理

灯丝是在不妨碍显示的极细钨丝蕊线上，涂覆上钡（Ba）、锶（Sr）、钙（Ca）的氧化物（三元碳酸盐），再以适当的张力安装在灯丝支架（固定端）与弹簧支架（可动端）之间，在两端加上规定的灯丝电压，使阴极温度达到 600^0C 左右而放射热电子。

栅极也是在不妨碍显示的原则下，将不锈钢等的薄板予以光刻蚀（PHOTO-ETHING）后成型的金属网格（MESH），在其上加上正电压，可加速并扩散自灯丝所放射出来的电子，将之导向阳极；相反地，如果加上负电压，则能拦阻游向阳极的电子，使阳极消光。

阳极是指在形成大致显示图案的石墨等导体上，依显示图案的形状印刷荧光粉，於其上加上正电压后，因前述栅极的作用而加速，扩散的电子将会互相冲击而激发荧光粉，使之发光。图 3.3 即表示其基本工作原理。发光色为绿色（峰值波长 505nm），低工作电压的氧化锌：锌(ZnO：Zn)荧光粉则是目前最被广为使用的荧光粉。

另外，通过改变荧光粉种类，可以获得自红橙色到蓝色的各种不同颜色。

3．2 VFD 的显示方式

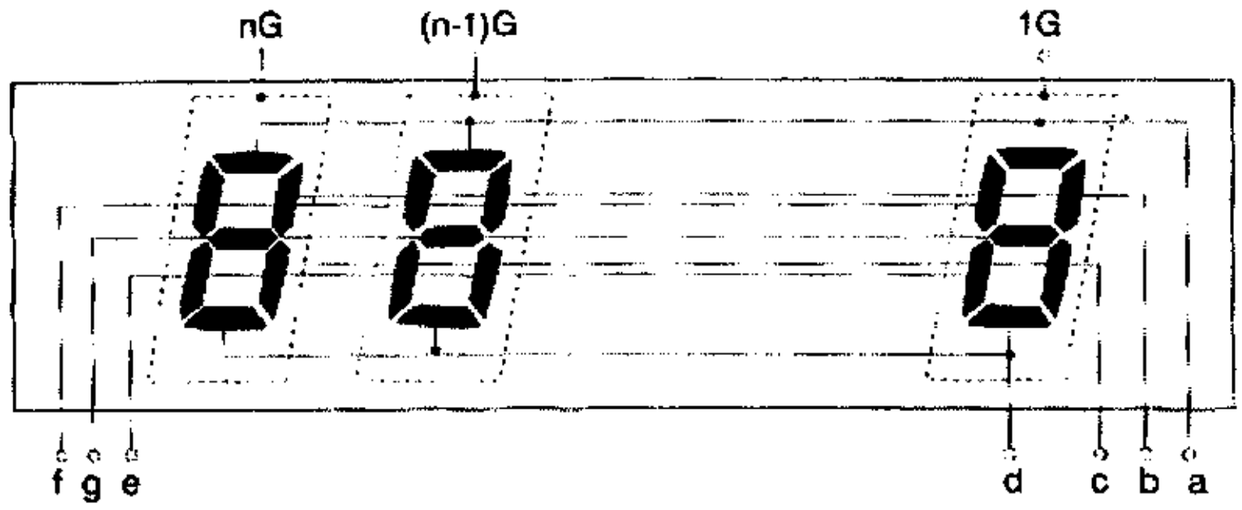


图 3.4　动态驱动方式荧光显示屏的电极连接

栅极和阳极的内部电极连接与导线的引出因驱动方式而异，在此做简单的说明。

图 3.4 所示是动态驱动方式荧光显示屏的基本电极连接图。从中可以清楚看到动态驱动是每个栅极各自独立引出，阳极则是每个栅极所对应的笔划共同连接、共同引出，因此即使位数多，阳极引出脚也无须随之增加。在多位数显示时，一边对各栅极加上栅极扫描电压（Gird-scan），同时也适时地对选择各阳极施加 ON（正）或 OFF（负）的脉冲电压，以快到肉眼无法觉察其间断的扫描速度，进行分时的动态驱动。

在关闭脉冲信号时，为加快振荡衰减，在输出线与栅极之间必须加上下拉电阻（PULL DOWN）。而在已施加电压（ON）的状态下，下拉电阻则是一种与显示屏并联的负载，会消耗无效电力，因此取值不能太小，反之，如果取值太大，容易产生振荡，或因尖峰电压（SPIKE NOISE）而有漏光的现象，通常以数 10KΩ（欧姆）左右最为合适。

图 3.5 是栅极（位数）与阳极（笔段）上脉冲信号的时序。在每个分离出来的栅极上，顺序施加配合图 3.5 的位数信号的脉冲电压，在阳极上则施加配合栅极扫描信号的脉冲电压（图 3.5 下方以“1234”所示的 4 位数）。

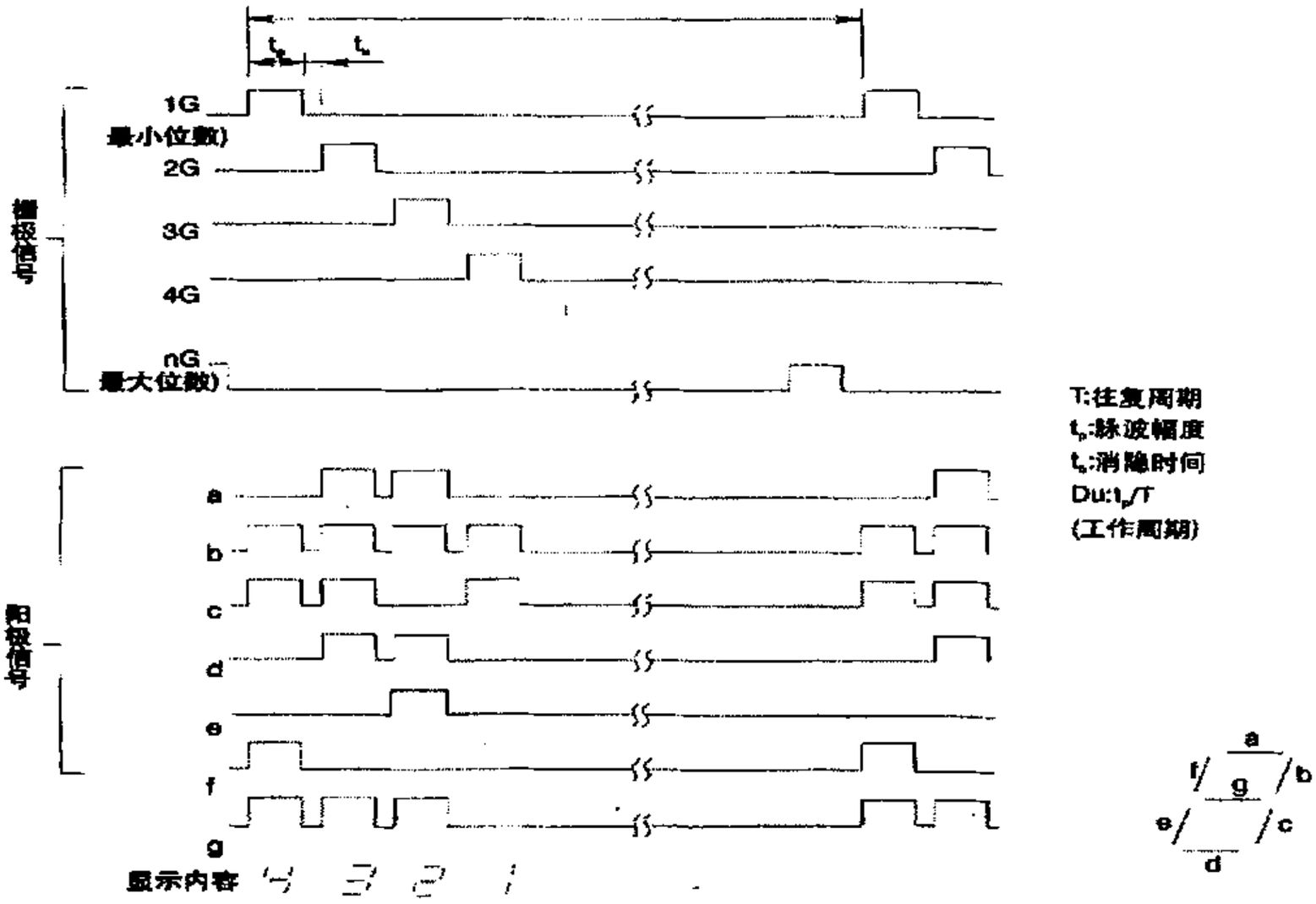


图 3.5　栅极（位数）与阳极（笔划）信号的时序

3．3 VFD 显示基本驱动电路

无论是动态或静态驱动，一般多采用 TTL 或 CMOS 数字电路作信号输出，也可采用直接驱动荧光显示屏的单片机或专用驱动芯片。

电路的构成可大致分为串联输入/并联输出及并联输入/并联输出两种方式。根据电源的供给方式则可分为正电源供电和负电源供电两种。

驱动器件可采用单个晶体管（TRANSISTOR）或电阻来构成，但为了简化电路一般使用各种专用驱动 IC。

电路的构成如图 3.6 所示

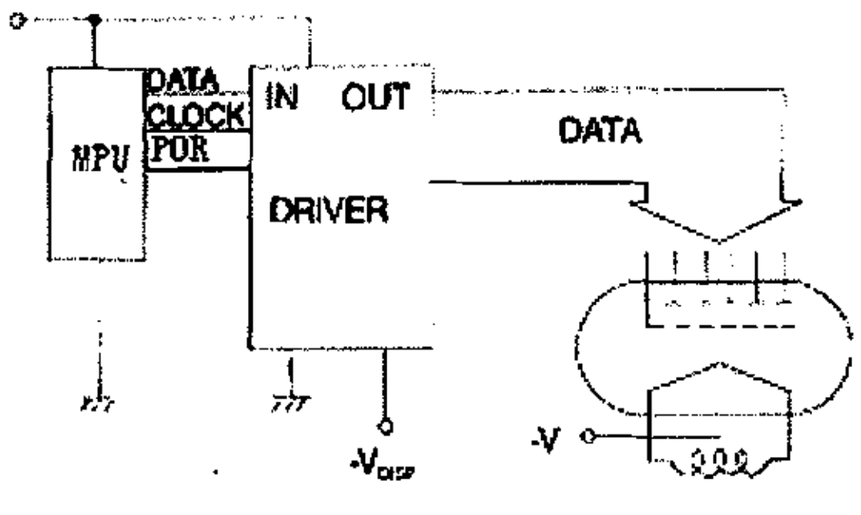


图 3.6　VFD 工作的驱动电路

3．4 本章小结

本章主要介绍了真空荧光显示屏（Vacuum Fluorescent Display）的结构及工作原理，并对 VFD 的显示方式和其基本驱动电路进行了详细的叙述。由于本论文的主要任务是设计 VFD 显示所必须的专用驱动芯片，所以理解 VFD 的工作原理及外围电路对于此芯片的设计来说是极其重要的。

第四章　VFD 驱动芯片的电路实现与功能仿真

4．1 VFD 专用驱动芯片的设计要求

VFD 显示控制驱动模块是在灯丝、栅极（GRID）、阳极（SEG）之间建立电场。由驱动电路循环地给栅极施加选择脉冲电压，同时，给阳极施加选择脉冲电压，以实现对荧光粉的激活。这种行扫描是逐行顺序进行的，循环一周期为一帧。驱动芯片可以通过接收 MPU 的控制来驱动 VFD 的显示。所以我们要设计的 VFD 专用驱动芯片应具有以下主要功能：

1） 以 8bit 串行输入方式为 MPU 提供接口；

2） RAM 与 ROM 的管理和控制，以便于 MPU 的访问；

3） 具备控制指令集便于 MPU 编程；

4） 完整的逻辑控制电路与时序发生器电路，以便控制各种显示功能；

5） 具有较高的驱动电压输出。

这块驱动芯片是专用于 16 位“米”字型显示屏，可以显示 64 个 ASCII 字符，如图 3.7 所示：

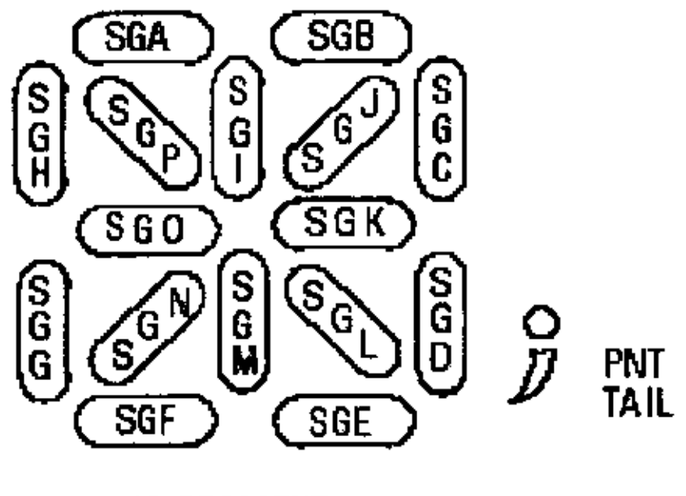


图 3.7　“米”字型显示块

芯片含有 16 个 GRID（栅）和 18 个 SEG（段）输出，通过接收串行数据信号和时钟信号来驱动 VFD 显示。内建 ASCII 字符库，以 ROM 形式设计，通过输入代码即可输出相应字符，其编码如图 3.8 所示：

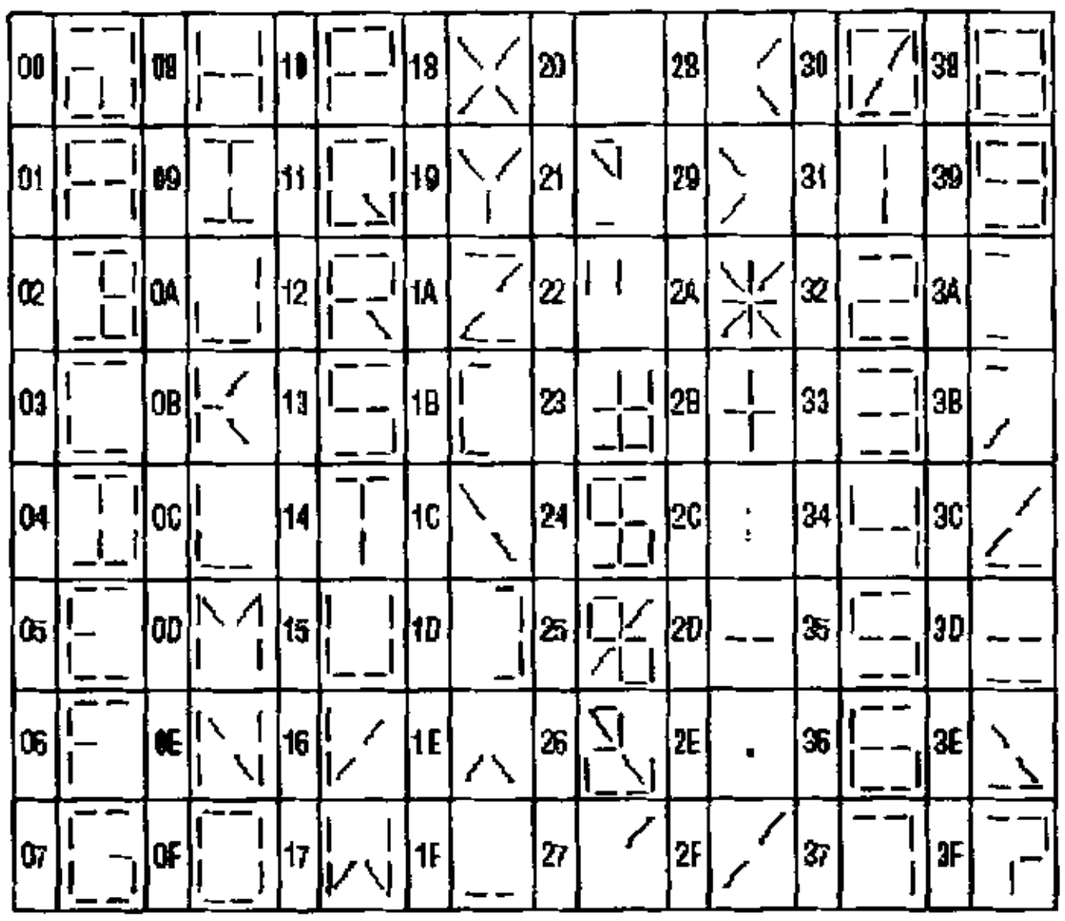


图 3.8　ROM 内字符编码

芯片采用 CMOS 工艺设计，具有较低的功耗。

4．2 总体设计框图

根据芯片的功能要求，我们把内部电路划分为图 4.1 所示的结构框图：

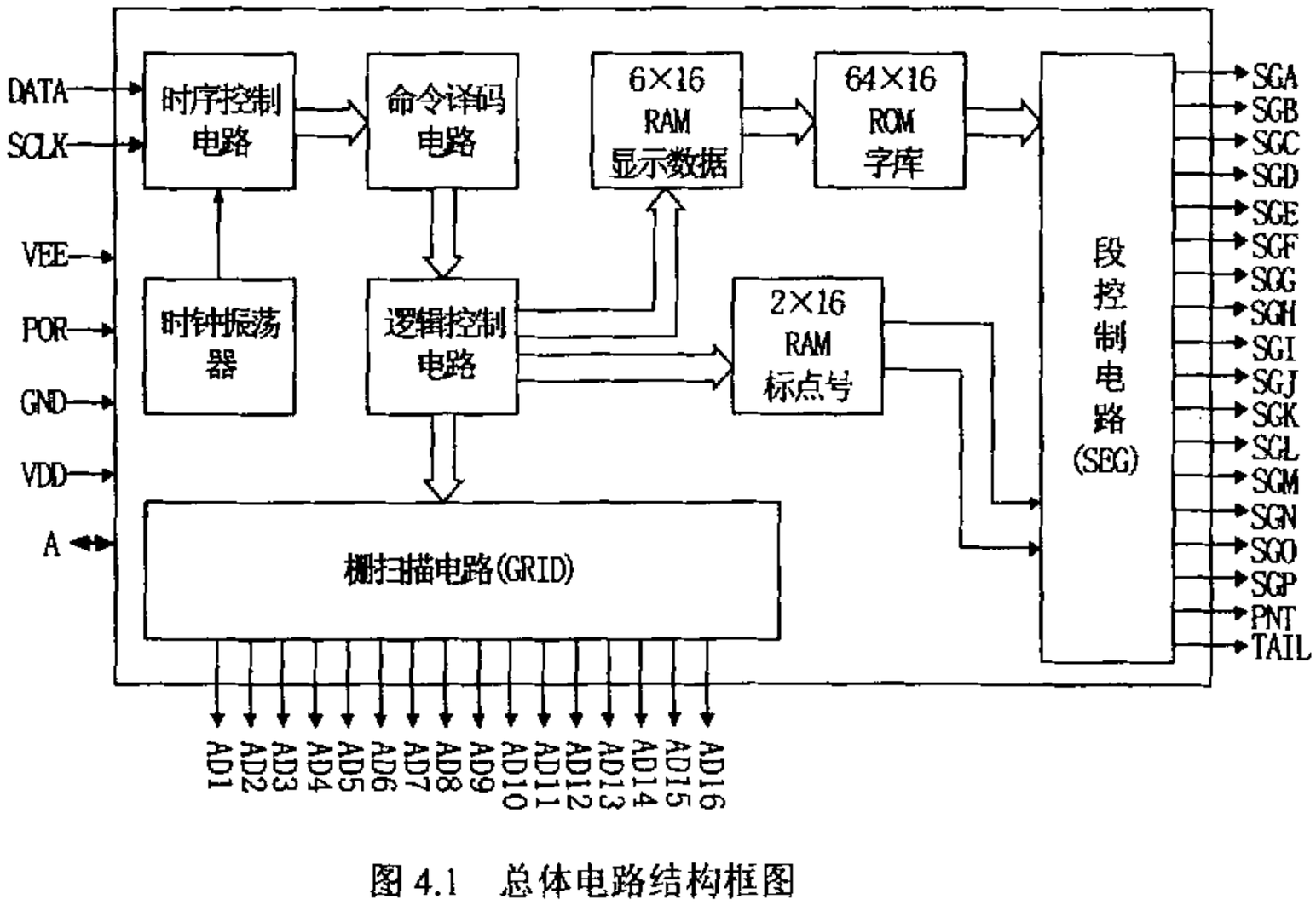


图 4.1　总体电路结构框图

本章研究的重点主要在于 VFD 驱动芯片的电路设计与仿真方法，并就 VFD 专用驱动芯片的关键模块电路：输入接口电路、时序控制电路、命令译码电路、逻辑控制电路、RAM 电路、ROM 电路的设计作详细的分析，从而完成芯片的前端设计。

4．3 输入接口电路设计

接口模块是专用驱动芯片与 MCU 通信的接口，数据交换的通道。MCU 通过它对芯片进行控制，向芯片中写入控制命令和显示数据。该模块有三个输入信号，串行数据信号（DATA），串行时钟信号（SCLK），复位信号（RST）。数据以字节（8-bit）为单位，高位在前，低位在后送入芯片，接收端在时钟的下降沿采样。该模块又可分为三个部分：串-并转换电路、延时控制电路、数据保持电路。

串-并转换电路由 8 个 D 触发器串联组成，构成了一个 8 位的移位寄存器，在同步时钟的控制下，将 8 位数据从高位到低位依次写进这 8 个 D 触发器。

延时控制电路主要由一个 8 进制计数器构成，当 8 位数据送进芯片后，在内部振荡产生的基准时钟配合下，锁存信号（LA）产生一个高脉冲，将读入移位寄存器的 8 位数据写进 8 个锁存器。同时命令译码有效（CMD_EN）信号也产生一个高电平信号，提供给命令译码模块。

数据保持电路由 8 个锁存器构成，在 LA 上升脉冲的控制下，将 8 位数据锁存，等待处理，实现了输入数据的串-并转换功能。

4．4 时序控制电路设计

该电路是实现显示输出的时序信号发生和控制中心，向整个芯片电路提供基准时钟。该驱动芯片的时序产生电路主要由以下几个电路模块组成：振荡产生电路、上电复位电路、基本时序产生电路等。

时序控制电路通过时序逻辑和组合逻辑产生多个时序信号，通过上电复位信号（POR）初始化整块芯片的时序，初始化芯片的工作状态；为其它电路模块提供 RST 复位信号；为命令译码电路提供一系列基准时钟；为 RAM 和 ROM 提供使能信号。

4．4．1 上电复位电路的设计

由于处于工作状态的芯片在突然断电时，芯片内部模块将以不确定的状态值保持，而这状态值将影响下一次上电时的正常工作，所以这之前有必要初始化内部各模块的状态值。

“上电复位”即 IC 上电后的初始化逻辑（简称“POR”），以保证所有内部数字逻辑的稳定性。常规做法是 IC 上电后，产生一固定宽度的“HIGH　PULSE”，至于其宽度的大小，则视 POR 所驱动的所有 GATE 时延的总和决定。由于 GATE 的时延常为 NS 级（由生产工艺决定），故一般设计时只要复位宽度在 US 级就足够了。

要达到以上目的，有两种方法：第一种为在断电后的一刹那进行初始化，另一种则是在上电时进行初始化。比较两种方法，后一种相对有效一些，因为在断电后到下一次上电存在一段时间，在这段时间里，内部各模块的状态值受外部环境影响而改变的概率大于零，所以这里采用上电复位电路。

要完成初始化工作，我们只需要一个能够输出在上电的一刹那的复位信号和相对稳定的工作信号。考虑到芯片的实际应用，我们采用了外接 POR 信号的方法，给芯片提供一定宽度的脉冲，并通过 RS 触发器保持，为芯片内部提供复位信号。

电路如图 4.2 所示：

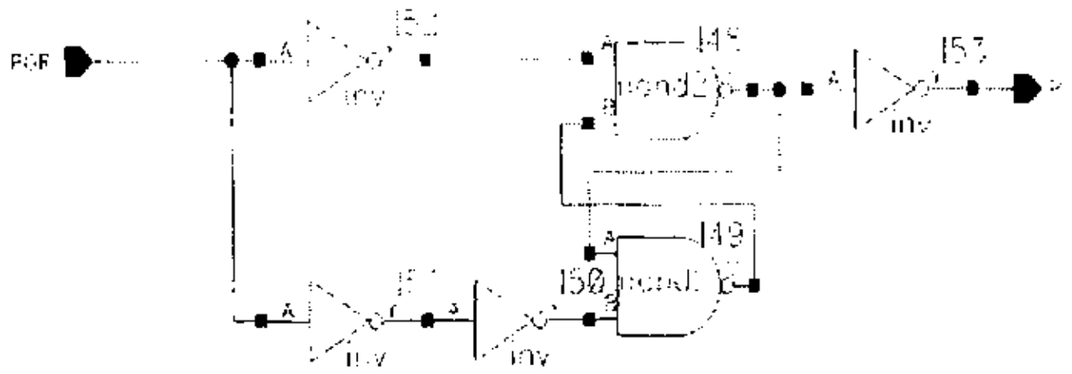


图 4.2　上电复位电路结构

芯片上电复位后，显示模式被设为 16 位，显示亮度被设为关闭，缓冲器地址被设为第 1 位，这些初始状态都通过复位信号控制相应电路模块来实现。

4．4．2 时钟振荡电路的设计

为了给整个电路提供一个基准时钟，以协调各部分电路的工作，并为命令译码电路提供信号源，我们需要一个周期性翻转的信号。除此之外，它还应具备与外部时钟相互切换的功能。

在参考电容充放电的工作原理和 CMOS 管域值电压的曲线特性的前提下，我们完全可以利用电容充放电的特性以及 CMOS 管域值电压与 CMOS 管导通的关系，设计出利用不断充放电的电容两端电压的变化使得门电路的输出电压不停的翻转的电路。其之所以可以循环变化，主要是由于电容两端电压的变化的连续性和门电路翻转的相对离散性相互作用的结果。

为了实现以上功能，我们需要设计一个循环变化的电路，带有使能端，且其输出信号可与外部信号相互切换。在 VLSI CMOS 设计中，常用到的振荡器有以下三种：RC 振荡器（RC Oscillator）、Ring 振荡器（Ring Oscillator）、Crystal 振荡器（Crystal Oscillator）。

它们的性能比较如下：

	优点	缺点
RC Oscillator	成本比较低，以现今技术而言，RC 均可 Built-in Chip	稳定度不佳，受过程（Process）的影响大，工作电压影响其频率
Ring Oscillator	工作频率随电压的影响不大，振荡电阻可以接在 IC 外部，也可以 Built-in Chip	电路相对复杂，布局尺寸面积较大，稳定度稍好一些
Crystal Oscillator	工作频率很准、稳定，仅与所选晶体器件的精度有关	成本比较高（相对于民用消费类产品），用于特殊要求或与 Clock 、Watch 、军工、

		通讯类产品合用

表 4.1　三种振荡器性能比较

由于该驱动芯片工作频率较低，频率受电压、电流影响小，所以我们采用 RC 振荡器，并把它做进芯片内部。振荡频率取决于 RC 的值，经验近似估算值为 T=2.2RC。

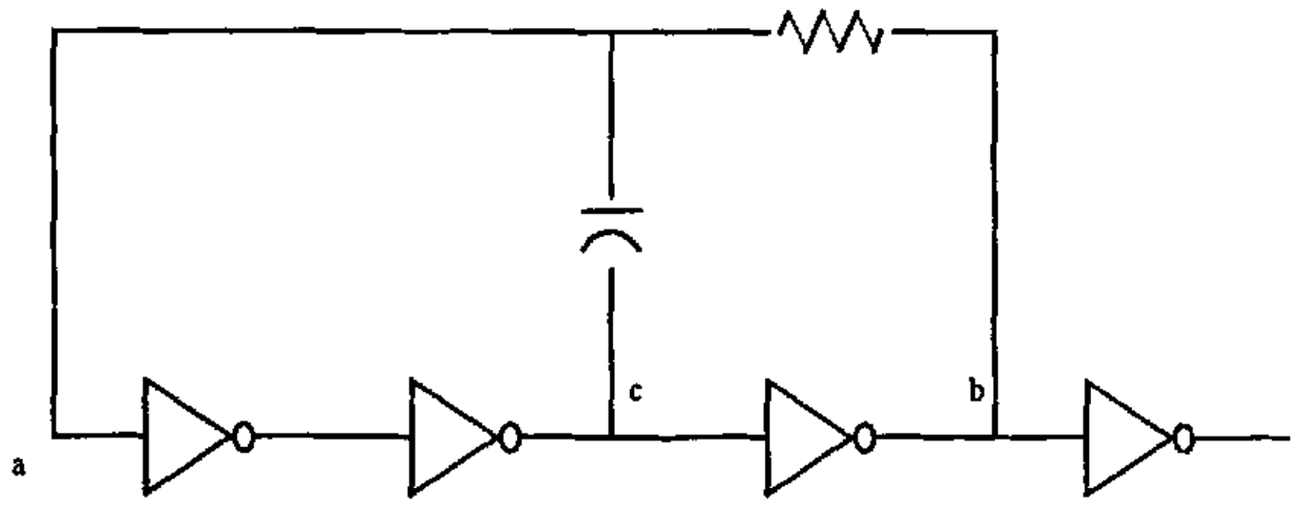


图 4.3　内部振荡器结构

芯片内部时钟振荡器工作原理如下：上电复位时，振荡器开始工作，此时 c 为高电平，b 为低电平，电容开始充电，这种状态称为第一暂稳态。随着充电的进行，a 点电势降低，结果使非门状态发生改变，由输出低电平变化到输出高电平，此时 c 为低电平，b 为高电平，电容反向充电，电路进入第二暂稳态。随着反向充电的进行，电容不断饱和，a 点电势随之升高，结果又使与非门状态发生改变，电路重新进入第一暂稳态。如此反复，在 c 端也就等价于输出端得到周期性的电平变化，起到了时钟振荡器的作用。

图 4.4 是 a，b，c 各端的工作波形：

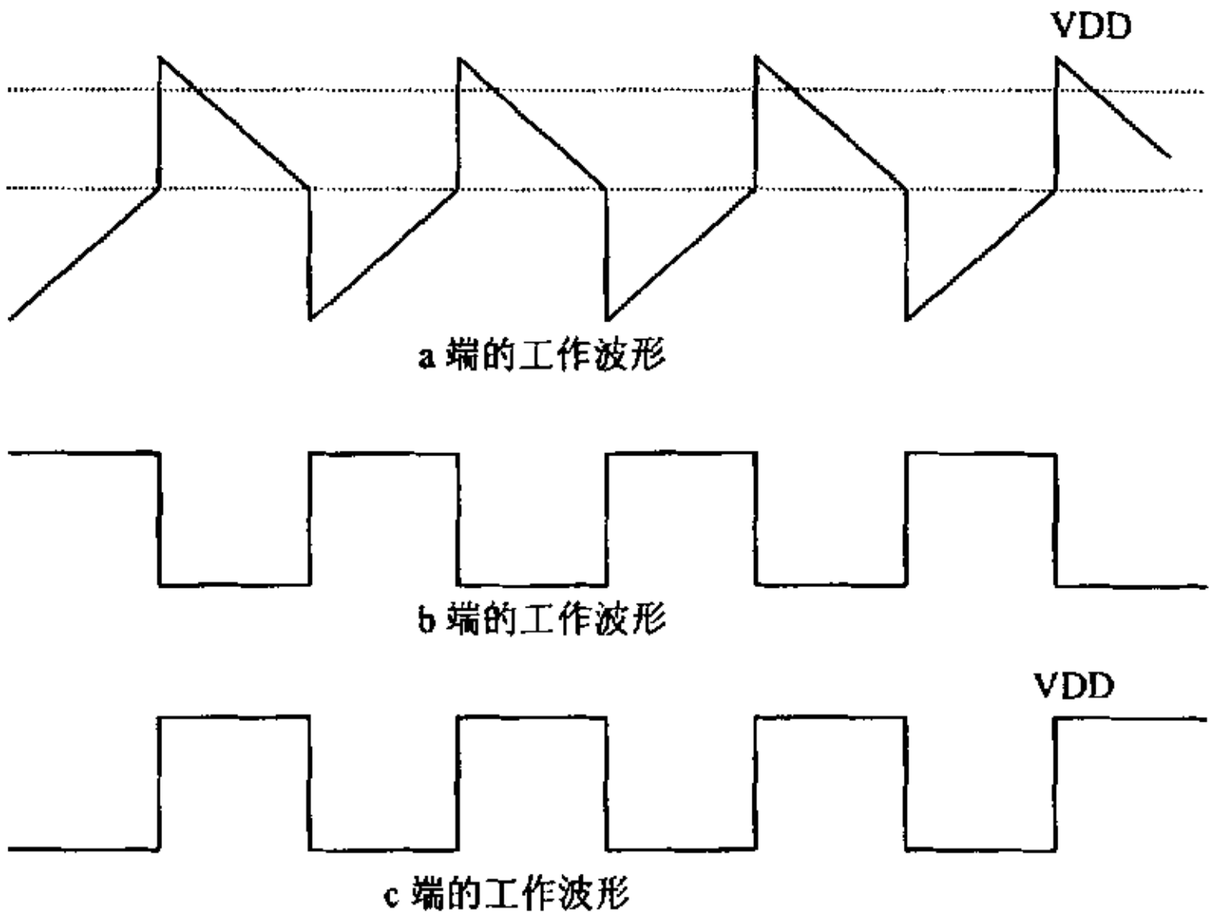


图 4.4　振荡器内各点的工作波形

芯片可以由两种方式来提供时钟信号，一种是外部时钟，应用于芯片的测试模式，另外一种就是采用上面提到的内部时钟振荡器，应用于芯片正常工作模式。如图 4.5 所示。在采用外部时钟振荡器时，信号由 A 接入，同时 ENB 接高电平，以封锁内部时钟振荡器产生的

信号干扰电路的工作；在采用内部时钟振荡器时，A 口作为输出口，输出内部时钟频率的 1/5，ENB 接低电平。内部时钟振荡器采用的是多谐振荡器。A 是一个双向口，通常处于输出状态，输出内部振荡频率的 1/5，当芯片处于测试模式时，A 用作输入端口。

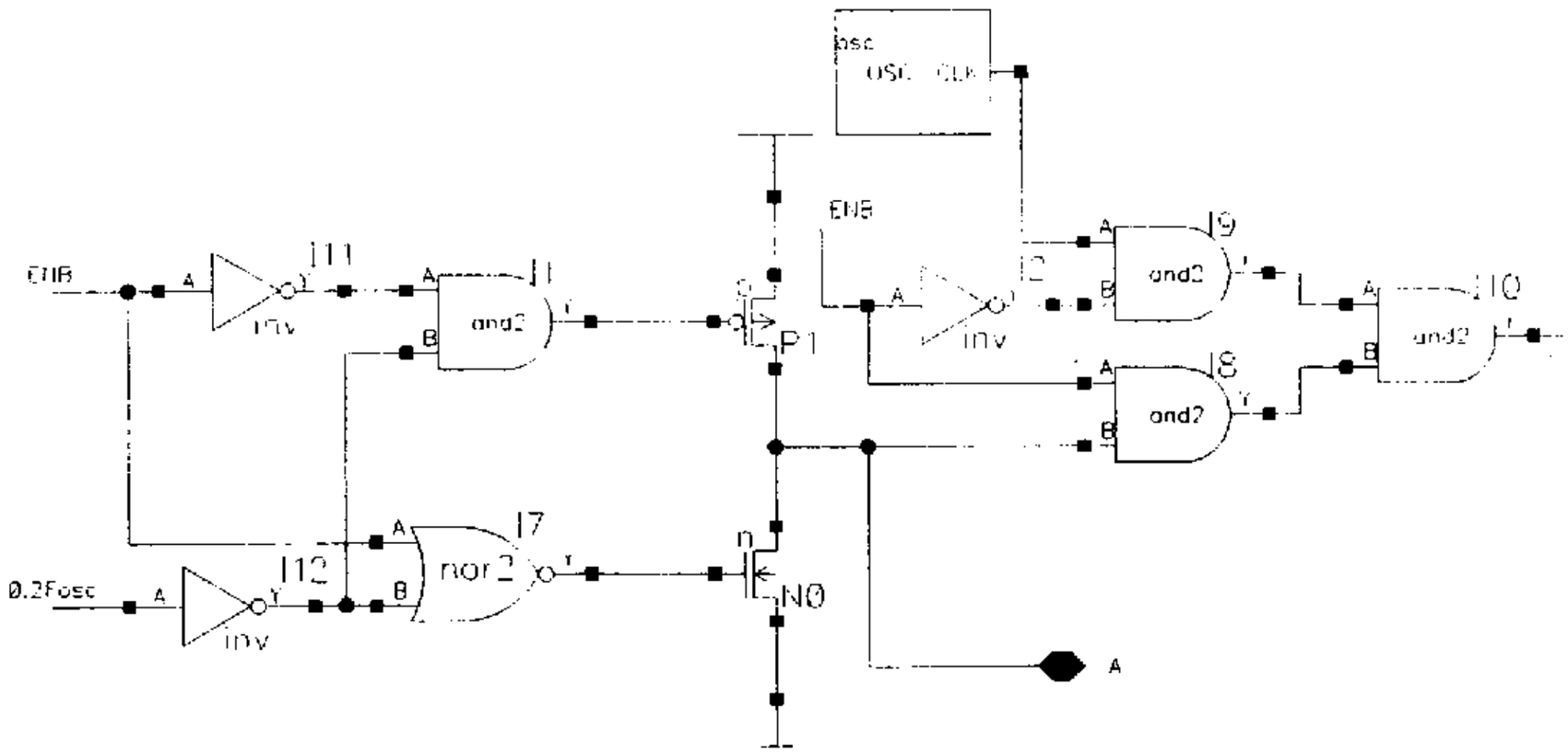


图 4.5　I/O 双向口电路结构

4．4．3 基本时序产生电路的设计

这部分电路主要由一个分频电路和一些组合逻辑电路组成。分频器的实质是规律性地重组输入信号的周期，以达到降低信号频率的目的，它对状态的顺序并不关心。虽然计数和分频是两个不同的概念，但从工作过程来看，两者都是在输入脉冲信号作用下完成若干个状态的循环运行，因此从广义来讲，分频器事实上是计数器在某专项功能领域的应用。电路如图 4.6 所示：

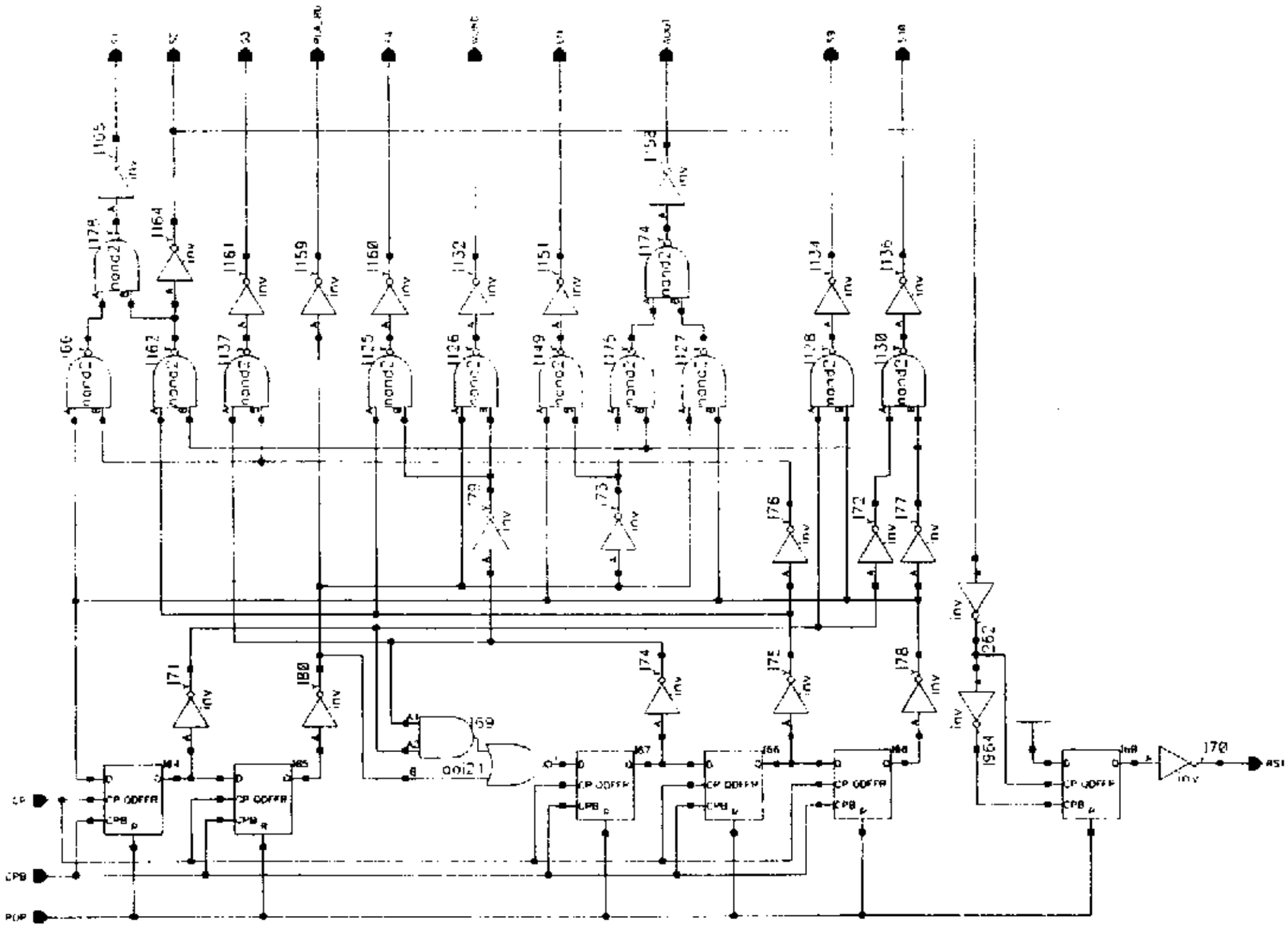


图 4.6　时序产生电路结构

该分频器电路采用的基本单元是标准的 D 触发器，考虑到同步电路在速度上优于异步电路，并且不易产生竞争-冒险问题，在电路中采用的是同步十进制计数器结构。

4．5 命令译码电路设计

该模块是专用芯片的控制中心，担负着区分数据/命令，解释输入指令的重要作用。它对 MCU 的各种输入进行识别，向相关模块发出控制信号，由后者完成指令实现芯片的相关功能。我们所设计的专用芯片共设置了 4 条命令。译码器的作用是将外部输入的命令字转化为具有一定宽度的高电位脉冲，从而触发相应的电路，使其状态发生改变，完成相对应的功能。译码器采用组合逻辑电路结构。来自 MCU 的指令代码通过专用芯片的接口模块电路产生 9 个输入信号，分别为 8 位命令字输入端、一个译码使能端。译码器根据 8 位数据的最高位来决定输入的是显示数据还是命令，如果最高位为“0”，表示输入的是显示数据，如果最高位为“1”，则表明输入的是命令，经过组合逻辑电路译码后分别产生 4 条命令。这 4 条命令分别为：地址设置命令，显示模式设置命令，显示亮度控制命令，设置测试模式。

地址设置命令用来设置要写入数据的存储单元地址，其命令格式如图 4.7 所示。图 4.7 中，bit0、bit1、bit2、bit3 四位用来设置存储单元地址，其范围为 0X00H～0X15H。在芯片刚接通电源时，地址的缺省设置为 0X00H。

MSB　　LSB

1	0	1	0	2^3	2^2	2^1	2^0

图 4.7　地址设置命令格式

显示模式设置命令用来设置芯片的显示模式。其命令格式如图 4.8 所示。图 4.8 中，bit0、bit1、bit2、bit3 四位决定了所选的显示模式。在芯片刚接通电源时，显示模式的缺省设置为亮 16 个字。执行该命令时显示会被强行关闭，要想恢复显示，必须执行一次显示控制命令，将显示打开。

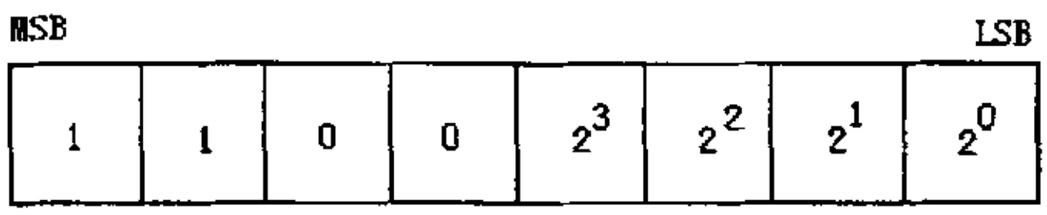


图 4.8　显示模式设置命令格式

显示亮度控制命令用来控制显示亮度。其命令格式如图 4.9 所示。图 4.9 中，bit0、bit1、bit2、bit3、bit4 五位用来控制栅极脉冲宽度。芯片是通过对栅极脉冲宽度的控制实现对显示亮度的控制。当芯片刚接通电源时，显示处于关状态。

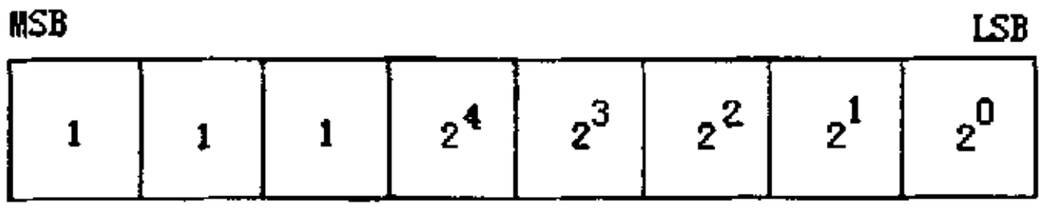


图 4.9　显示亮度控制命令格式

设置测试模式只供测试使用，芯片工作在该模式时，内部振荡器被屏蔽掉，由 A 口输入时钟信号。其命令格式如图 4.10 所示。图 4.10 中，bit0、bit1、bit2、bit3 都是无关位，bit4 决定了测试模式的种类。

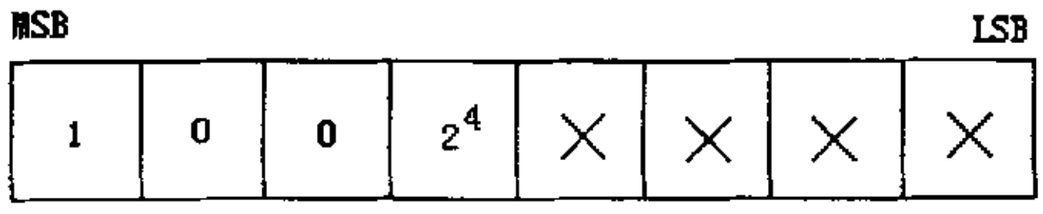


图 4.10　设置测试模式格式

命令译码部分电路如图 4.11 所示：

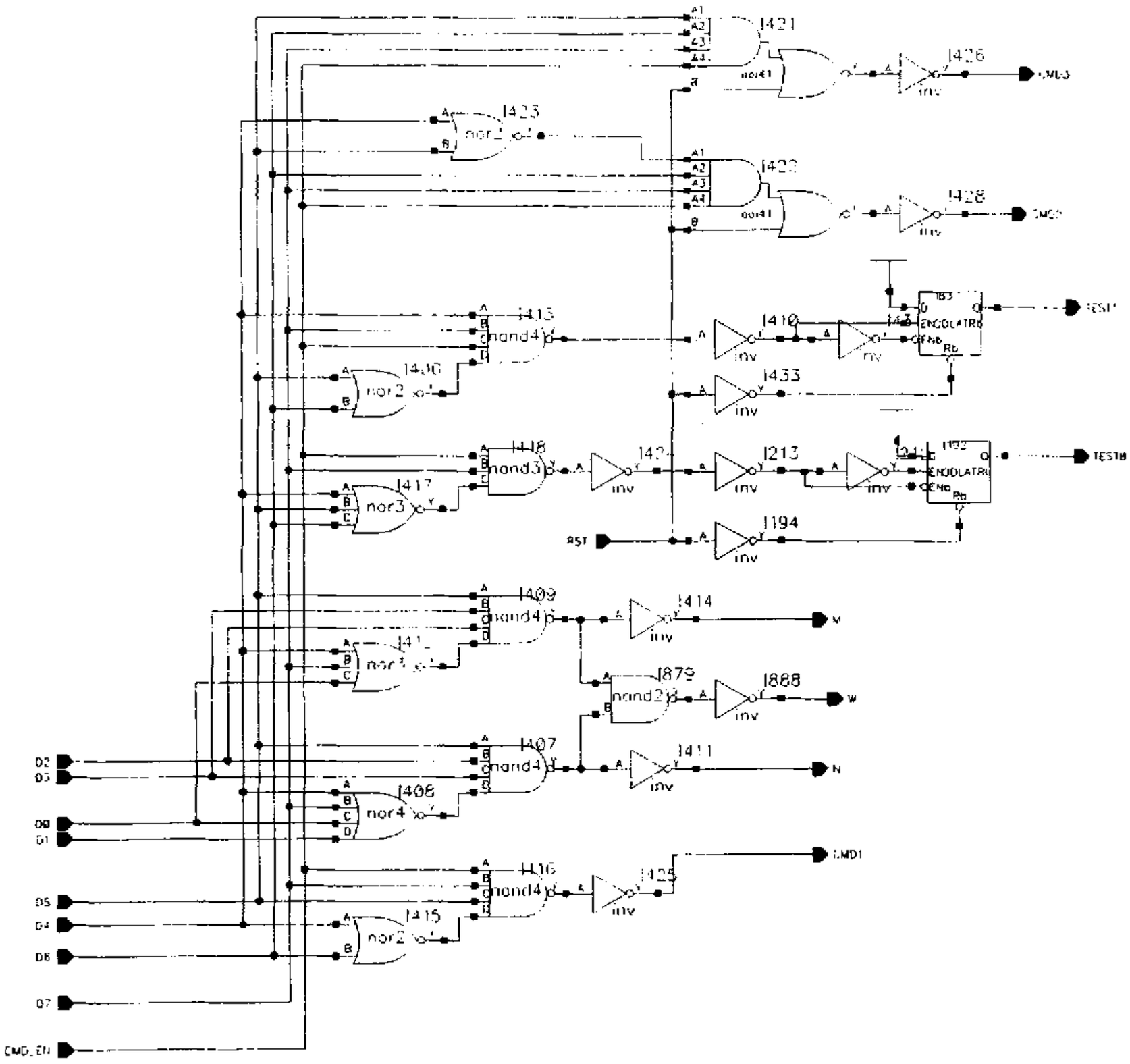


图 4.11　命令译码电路结构

CMD_En 是译码使能信号，它由输入接口电路模块产生，高低电平切换频率较高，高电平有效，当 D7 为“0”时，组合逻辑中的与非门全部锁住，译码电路不工作，输入数据为显示数据；当 D7 为“1”时，译码电路工作，若 bit6 ~ bit4 为“010”，则对应的 CMD1 产生 1 个高脉冲，向逻辑控制模块发出控制信号。同理，在不同命令的控制下，CMD2，CMD3，TEST 均会向逻辑控制模块发出高脉冲控制信号。

需要说明的是，在该驱动芯片对应的“米”字型显示屏中，每块栅上“逗号”和“小数点”的显示方式与其它段的显示方式是不同的。SGA ~ SGP 的显示内容是直接由芯片的 DATA 口输入的数据送进缓冲器，经过译码后从 ROM 读出，而这两点的显示是由命令译码电路对输入的数据进行判别后送入另一个缓冲器，然后直接经过锁存电路锁存输出。这主要体现在电路中的 point 0 和 point 1 两根信号线，当输入数据为“2C”或“2E”时，point 0 或 point 1 经译码电路变高电平，送入缓冲器，并且此时，RAM 的写入地址不会递增，所以芯片输出内容显示在前一数据所对应的栅上。这在缓冲器模块的设计中将会详述。

4．6 逻辑控制电路设计

该模块根据从命令译码模块接收到的命令信号、从输入接口模块接收到的数据信号、从

时序控制模块接收到的时序信号，在逻辑上和时序上对整个电路进行全局的控制，它可以说是该芯片的控制中心。主要由地址设置命令电路、显示模式设置命令电路、显示亮度控制命令电路构成。

4．6．1 地址设置命令电路的设计

图 4.12 是一个具有置数功能的十六进制计数器，它的主要功能是写 RAM 地址指针，在计数时钟控制下，它以步进的方式从 RAM 首地址指向末地址。它的工作状态主要由 CMD1 和 bit0～bit3 决定。当无地址设置命令输入，即 CMD1 保持为低电平时，四个置数口全部锁闭，并且在组合逻辑电路的控制下，四个 D 触发器全部清零，计数器停止工作；当接收到地址设置命令，即 CMD1 出现一个高脉冲时，置数口开启，bit0～bit3 的预设值被读入四个 D 触发器，然后在命令译码模块发送的写 RAM 地址计数器时钟 W 配合下，计数器开始工作，从置数值开始递增计数，显示数据也相应被写入该指针指定的 RAM 地址。

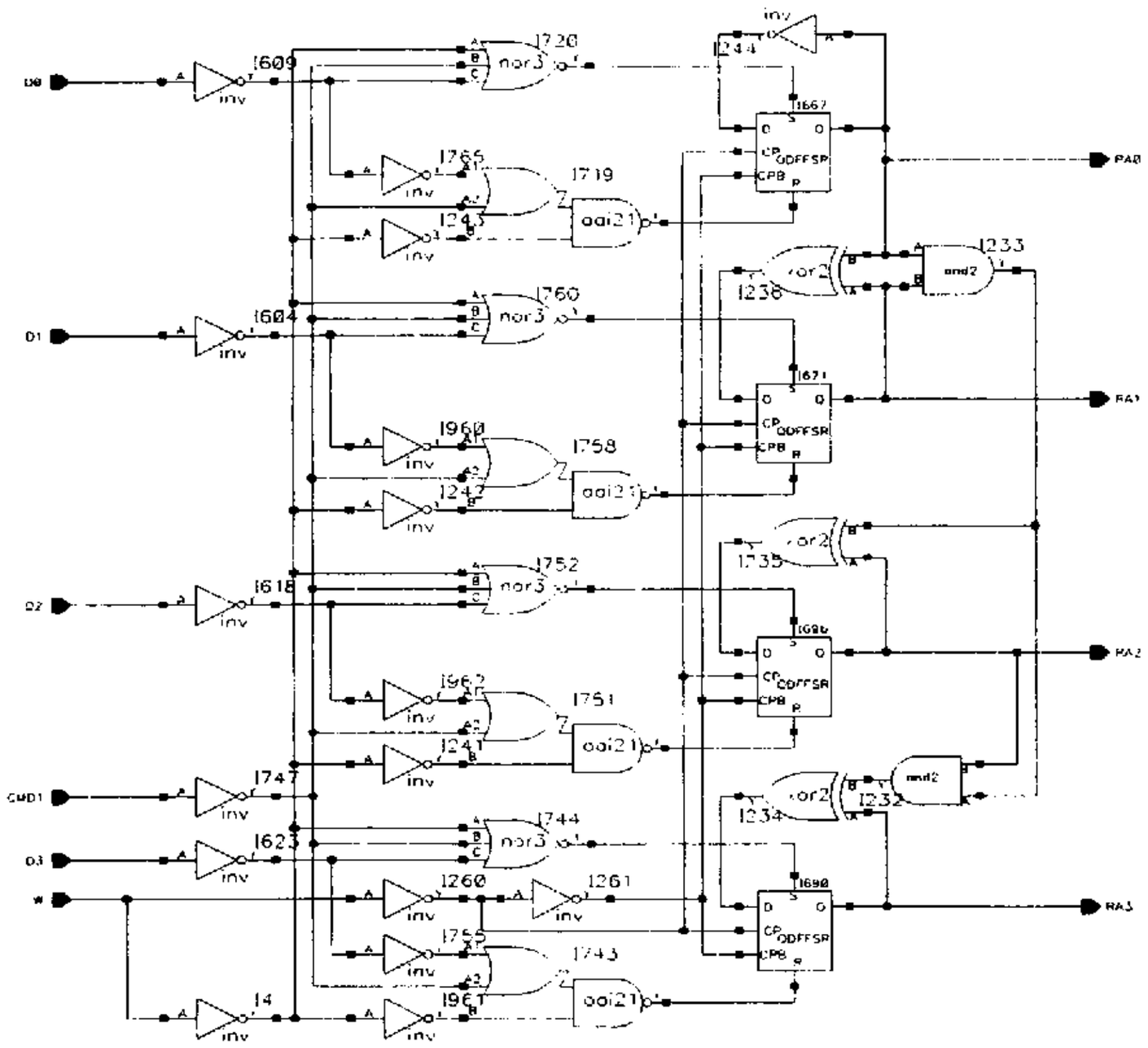


图 4.12　地址设置命令电路

RAM 地址与栅的具体对应关系如表 4.2 所示：

目标栅格	设置地址（bit3～bit0）	目标栅格	设置地址（bit3～bit0）
AD1	15（1111）	AD9	7（0111）
AD2	0（0000）	AD10	8（1000）
AD3	1（0001）	AD11	9（1001）
AD4	2（0010）	AD12	10（1010）
AD5	3（0011）	AD13	11（1011）

AD6	4（0100）	AD14	12（1100）
AD7	5（0101）	AD15	13（1101）
AD8	6（0110）	AD16	14（1110）

表 4.2　RAM 地址与栅的具体对应关系表

RAM 读写地址的切换功能是由另一个十六进制的计数器完成的，它的主要功能有两个：一是读 RAM 地址指针，它从首地址开始向预设末地址循环计数；二是栅扫描地址计数器，为栅扫描模块提供地址信号，这在后面还会继续提到。

需要说明的是，在设计中，写 RAM 地址指针和读 RAM 地址指针都是通过 RA3～RA0 口送入 RAM 模块的，它的切换过程是通过 8 个三态门，分 4 组，两两并联实现的，如图 4.13 所示。

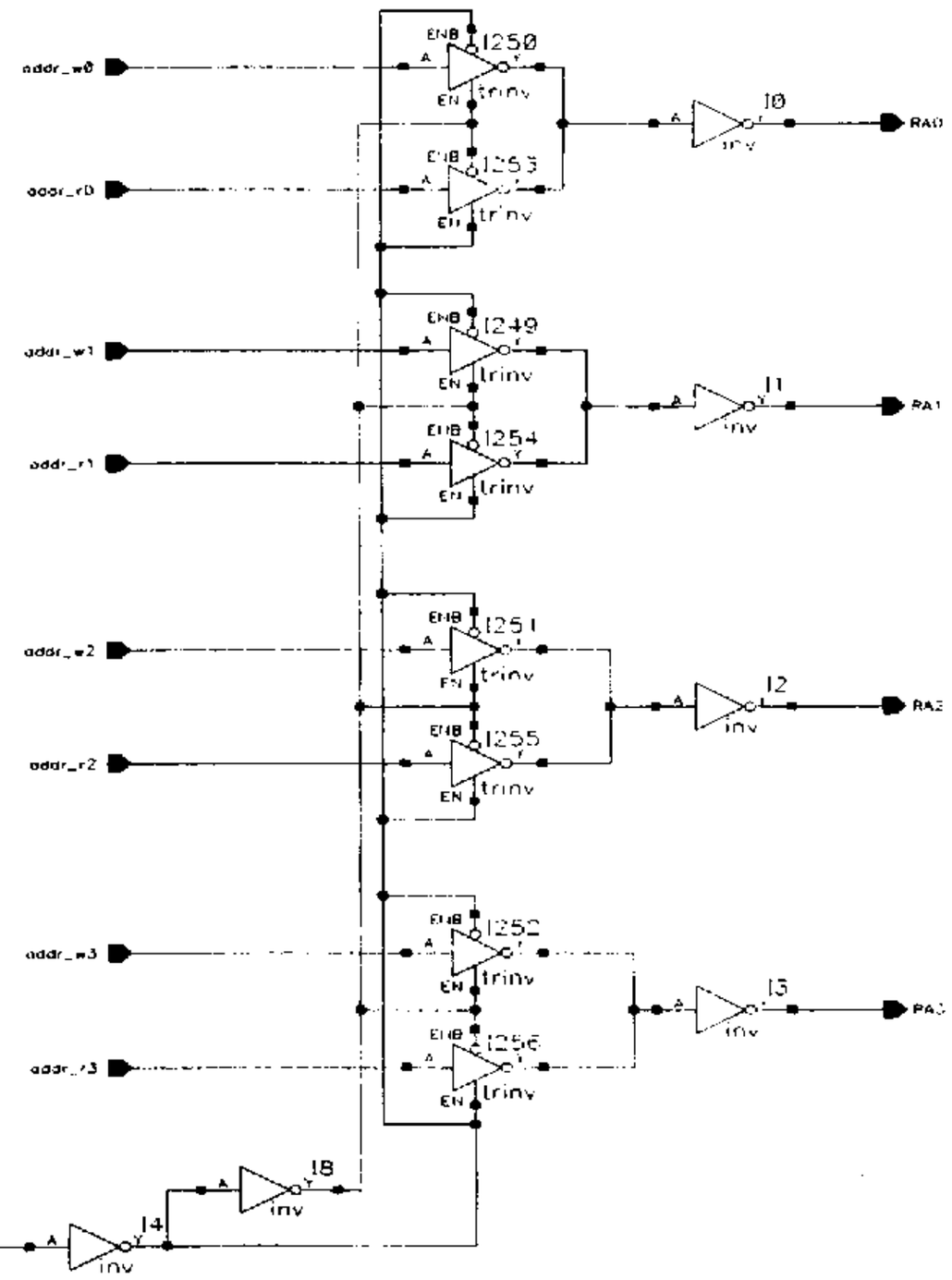


图 4.13　读 / 写转换电路结构

4．6．2 显示模式设置命令电路的设计

该部分电路如图 4.14 所示，它主要由减法电路和比较电路组成。它的主要功能是执行显示模式设置命令，工作状态主要由 CMD2 和 bit0～bit3 决定，显示位数与设置值的对应关系如表 4.3 所示。

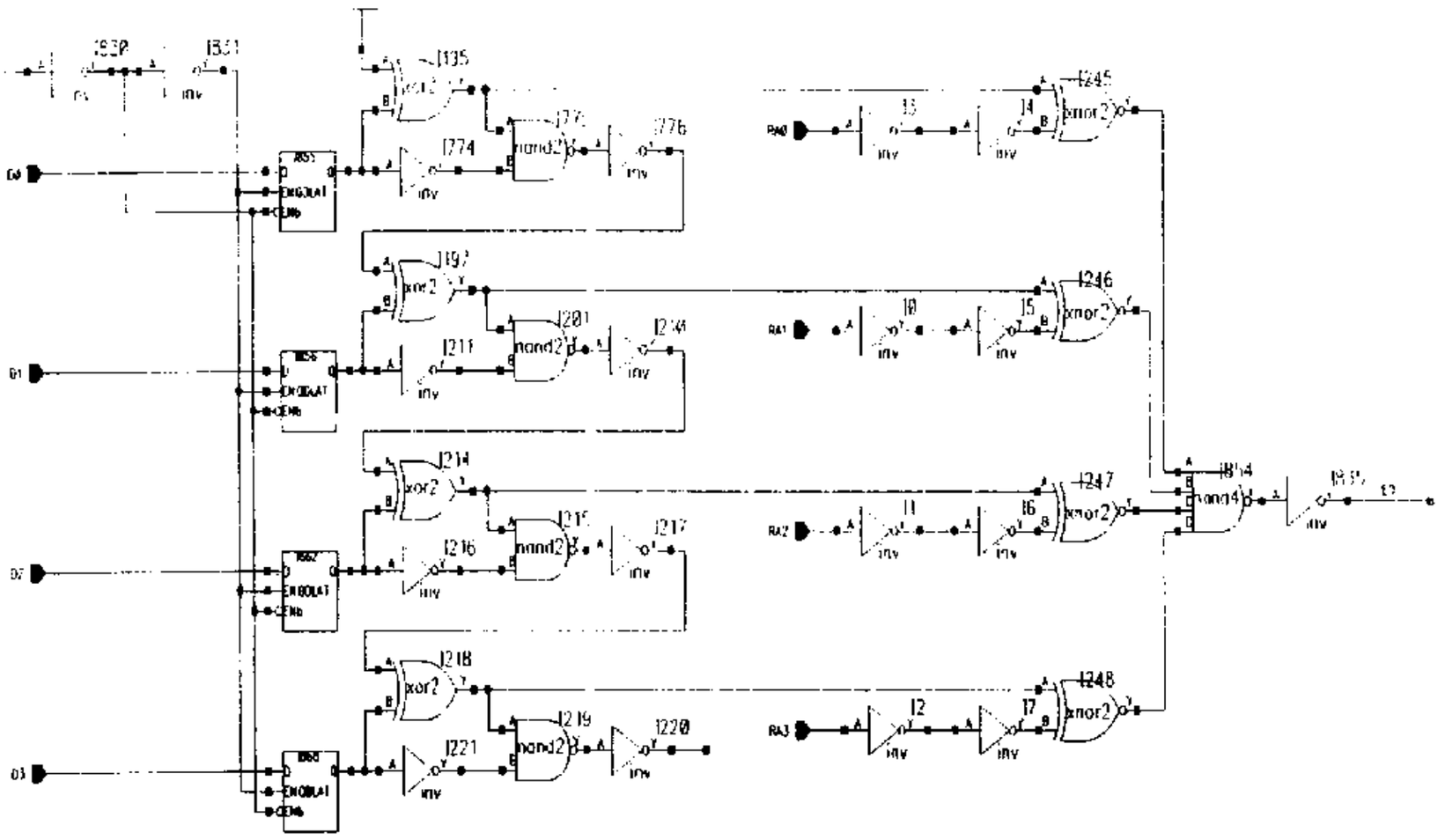


图 4.14　比较判决电路结构

显示位数	设置地址（bit3～bit0）	显示位数	设置地址（bit3～bit0）
1	1（0001）	9	9（1001）
2	2（0010）	10	10（1010）
3	3（0011）	11	11（1011）
4	4（0100）	12	12（1100）
5	5（0101）	13	13（1101）
6	6（0110）	14	14（1110）
7	7（0111）	15	15（1111）
8	8（1000）	16	0（0000）

表 4.3　显示位数与设置值的具体对应关系表

当无显示模式设置命令时，即 CMD2 为低电平时，四个锁存器的使能信号无效，bit0～bit3 不能通过锁存器进入比较电路；当接收到显示模式设置命令，即 CMD2 为高电平时，四个锁存器开启，此时，bit0～bit3 进入减法电路，被减一后，四位数据进入比较电路与写 RAM 地址进行判决，如果相等，则 EQ 产生一个高电平，使写 RAM 地址计数器复位，重新计数，如果不等，则写 RAM 地址计数器继续进行计数操作。

因为地址设置命令设置的地址不能大于在显示模式设置命令里设置的要求显示的栅的数目，所以必须经过比较电路进行判决控制。而减法电路的作用是调整显示模式设置命令的设置值，以与写 RAM 地址值比较。例如，根据上面命令译码器模块中对各条指令的介绍，如果显示模式设置为亮 8 块栅，则 bit3～bit0 为“1000”，此时写 RAM 地址在计数过程中绝不可能超过 8，当到达第九块栅时，即写 RAM 地址值为“0111”，必须与显示模式设置中的“1000”相比较，因为每次比较时，它们的值都相差 1，所以必须经过减法电路进行减 1 操作后才能进行比较。

地址设置命令和显示模式设置命令部分的电路仿真结果如图 4.15 所示：

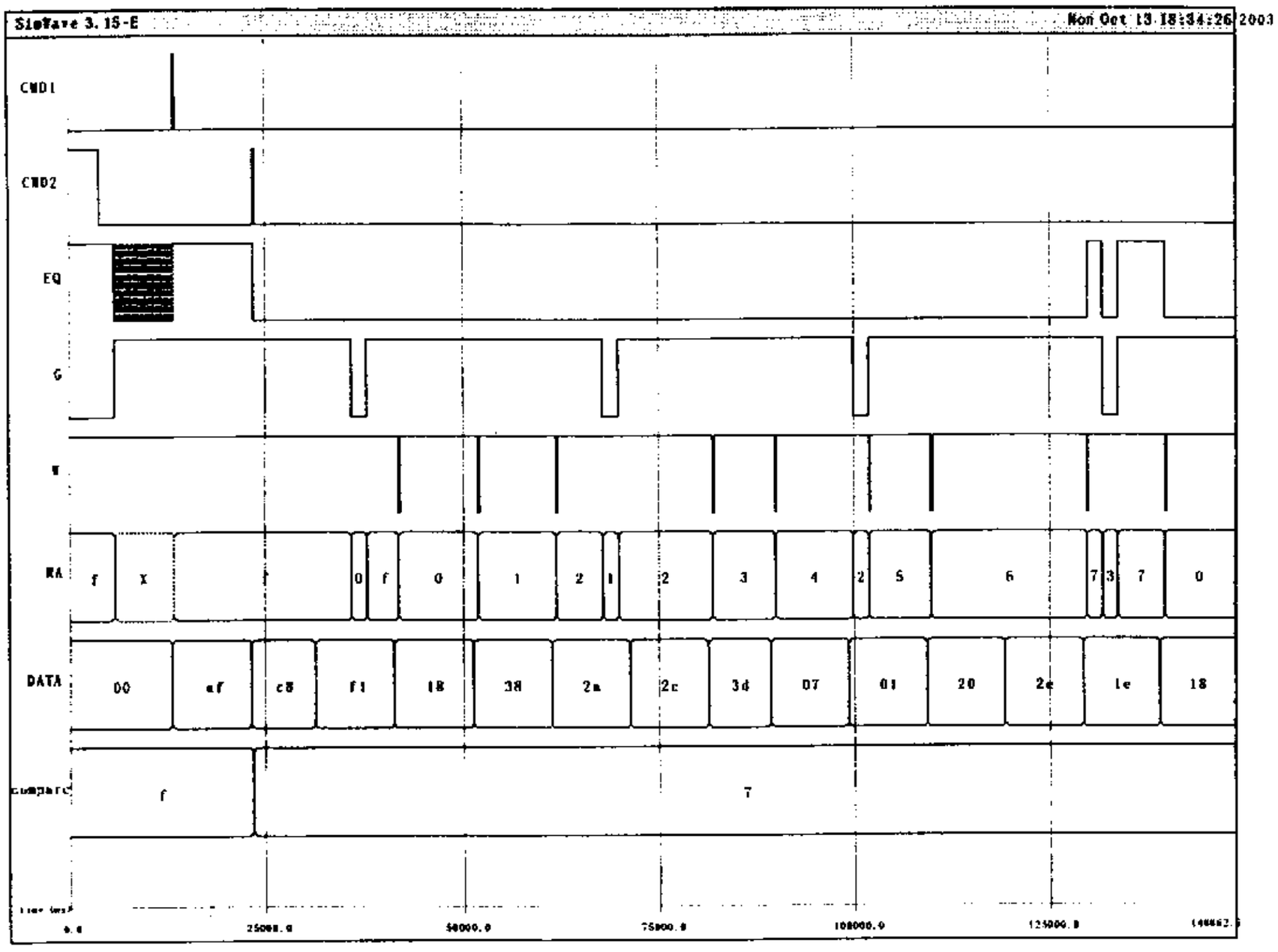


图 4.15　地址设置和显示模式设置电路 Verilog-XL 仿真结果

4．6．3 显示亮度控制命令电路的设计

图 4.16 是一个具有置数功能的三十二进制计数器，它的主要功能是调节显示亮度。根据 VFD 屏的显示原理可以知道，栅上的高电平持续时间越长，则从灯丝上吸引的电子越多，打到荧光粉上的电子也越多，显示就越亮，所以可以通过控制栅高电平的持续时间来调节显示的亮度。

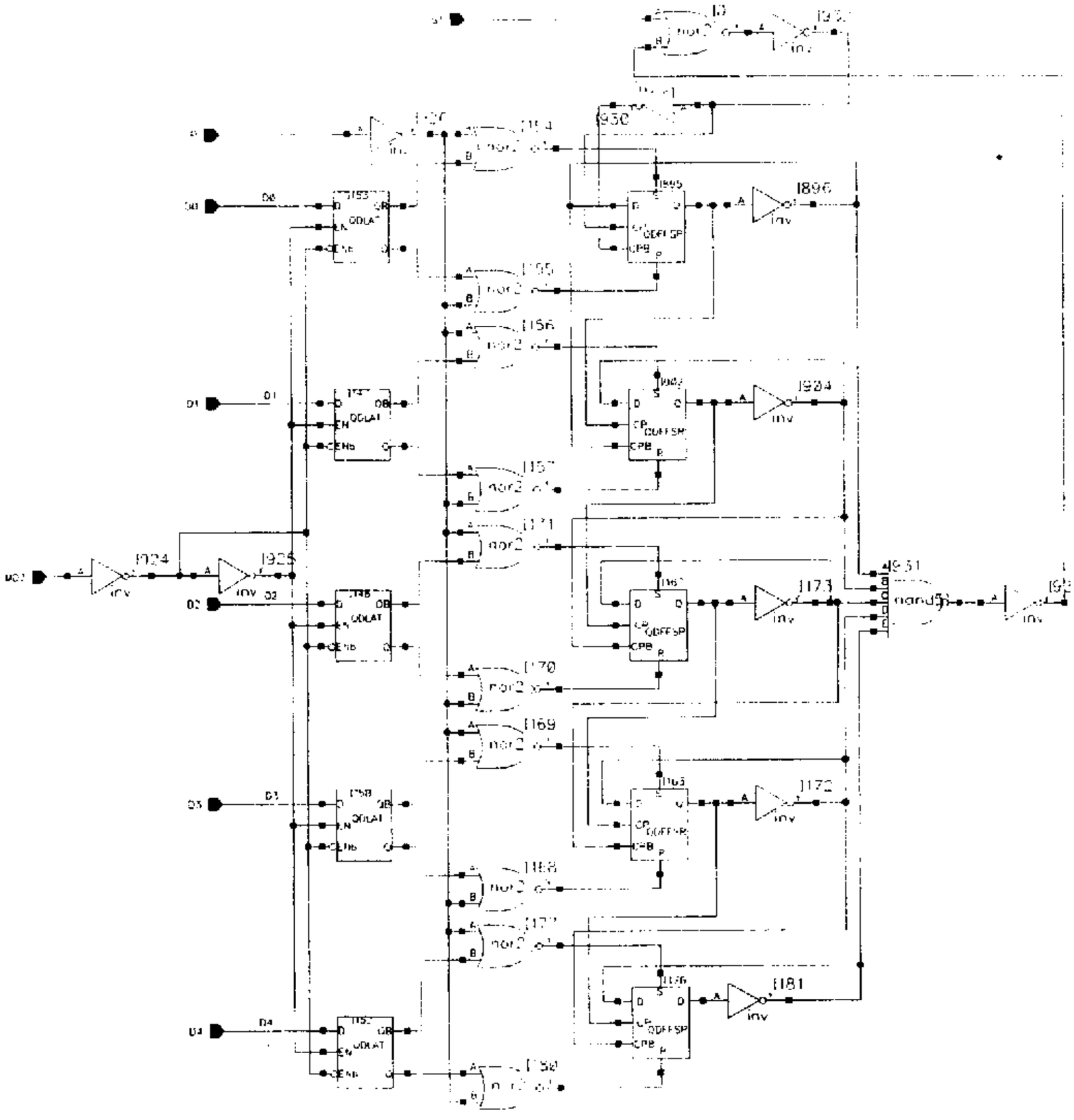


图 4.16　显示亮度调节电路结构

当有显示亮度控制命令送入时，CMD3 产生一个高脉冲，五个锁存器全部打开，把 bit4 ~ bit0 都置入五个 D 触发器中，当 CMD3 回到低电平后，bit4 ~ bit0 就一直被存储在锁存器中，同时信号 P 来个高脉冲，打开置数功能，为计数器设初值，直至有新的命令写入。而计数器就在设置的初值基础上进行计数，当计到 1F 时，五输入与非门的输出 LA 变为“0”，锁住了计数器的时钟信号，计数停止，同时，它使 SEG_DUTY 信号也由“1”变为“0”，此时段扫描模块的使能信号被关闭，所以段输出由高电平变为低电平，实现了对亮度的调节。扫描完一块栅以后，P 信号又将给出一个高脉冲，重新置数，计数器将开始第二个工作周期。

仿真结果如图 4.17 所示：

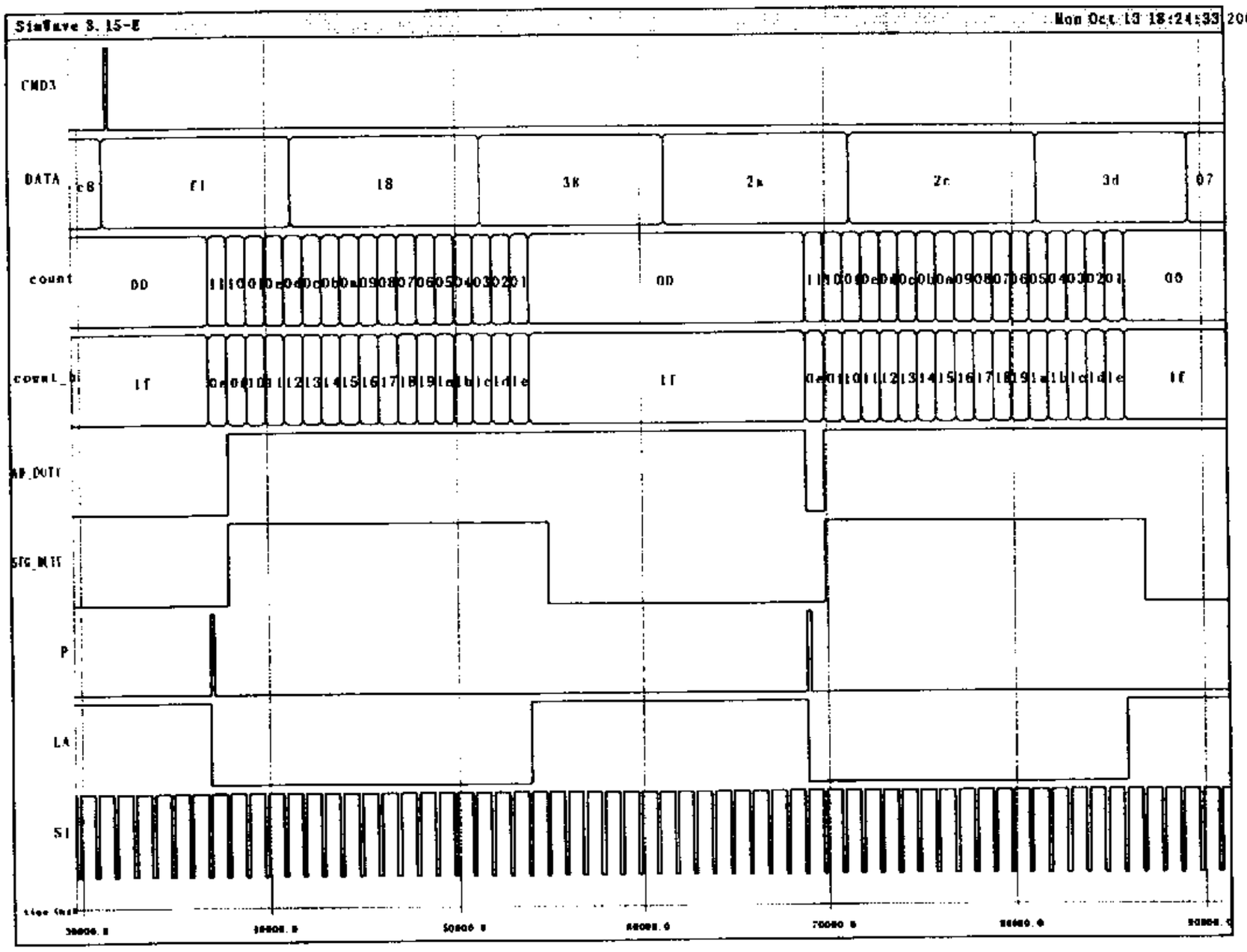


图 4.17　显示亮度控制电路 Verilog-XL 仿真结果

4．7 存储器电路设计

MOS 存储器最主要的产品是随机存取存储器（RAM），RAM 又分为两类：一类是动态随机存取存储器（DRAM），它是靠电容存储信息，即动态存储方式，单元面积小，易于提高集成度。由于 DRAM 集成度高，功耗低，适于做计算机的内存。DRAM 的缺点是信息不能长期保存，需要定期刷新。另一类 RAM 是静态随机存取存储器（SRAM），它是靠双稳态电路存储信息，是静态存储方式，只要不断电，信息可以一直保存，不需要刷新。SRAM 工作速度快，适合于作高速缓冲存储器。SRAM 的缺点主要是单元面积大，集成度不如 DRAM 容易提高，并且有较大的功耗。

MOS 存储器还有一类是只读存储器（ROM）。ROM 的存储信息一经存入可以长期保持，而 RAM 的存储信息在断电后就不存在了。

4．7．1 缓冲器（SRAM）电路的设计

SRAM 是静态存储方式，以双稳态电路作为存储单元，不需要刷新，外围电路简单，而且工作速度较快，所以在缓冲器的设计中采用了 SRAM 技术。

下面来分析一下 SRAM 单元的工作原理：

SRAM 是靠双稳态电路存储信息。一个 SRAM 存储单元由 6 个 MOS 管组成，构成双稳态电路要 4 个 MOS 管，即 2 个反相器交叉耦合构成，再加上控制单元选择的 2 个门管。下图是一个 CMOS 6 管单元电路。由于双稳态电路有 2 个引出结点，SRAM 单元有 2 个门

管，有 1 根字线 WL，2 根位线 BL 和 $\overline{BL}$，另外 SRAM 单元需要电源线和地线。SRAM 单元不仅所用元件比 DRAM 单元多，而且连线数目也多。

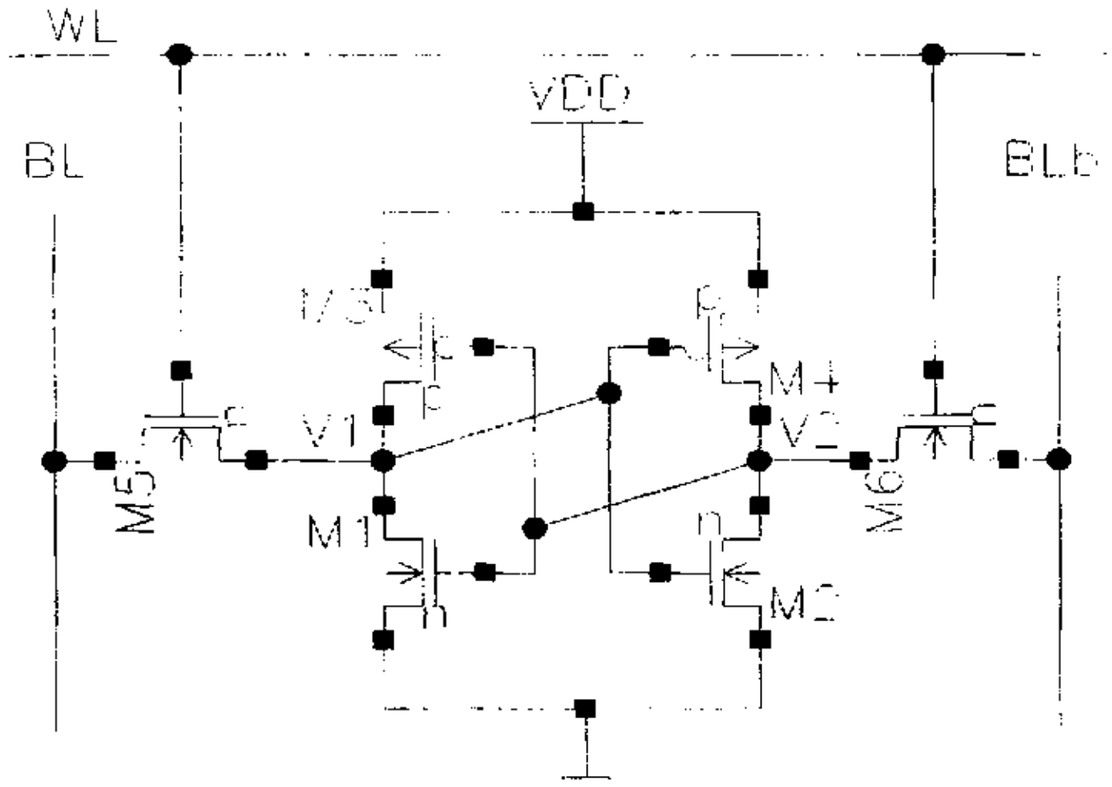


图 4.18　CMOS SRAM 单元电路

SRAM 单元在保持状态时字线 WL 为低电平，2 个门管 M_5 和 M_6 都截止。若单元原来存“0”，则 $V_1=0$、$V_2=V_{OH}=V_{DD}$。V_2 的高电平使 M_1 导通、M_3 截止，维持 $V_1=0$；另一方面，V_1 的低电平使 M_2 截止、M_4 导通，维持 $V_2=V_{DD}$。若单元存在“1”，则 $V_1=V_{DD}$、$V_2=0$，这种状态同样可以长期维持。

进行读操作时，选中单元的字线为高电平，两个门管都导通，把单元的存储结点 V_1 和 V_2 连接到位线 BL 和 $\overline{BL}$ 上。位线上有公共负载管对位线充电，使 BL 和 $\overline{BL}$ 在读操作前都被预充到高电平。若单元存“1”，位线 $\overline{BL}$ 通过导通的 M_6 和 M_2 放电，而 BL 保持为高电平，因而在位线 BL 和 $\overline{BL}$ 上得到正向的电位差，即

$$\Delta V = V_{BL} - V_{\overline{BL}} > 0 \ 。$$

若单元存“0”，则位线 BL 通过单元中导通的 M_5 和 M_1 放电，而位线 $\overline{BL}$ 保持预充的高电平。这样在两侧位线上得到一个反向的电位差。

$$\Delta V = V_{BL} - V_{\overline{BL}} < 0 \ 。$$

由于单元的尺寸很小，位线通过单元管放电的速度很慢，为了提高读出速度，只要位线上建立起一定的信号差就可以了，而不必等到一侧位线下降到低电平。通过列译码器控制的列开关，把选中单元位线读出的微小信号差 ΔV 送到公共数据线，再通过公共数据线（I/O 线）送到读出放大器，把微小的信号差放大为合格的高低电平，最后通过缓冲器转换成单端

信号输出。

在写操作时，通过行译码器使选中的字线为高电平，使单元门管导通，再通过列译码器使选中单元的位线与公共数据线接通。外部送入的数据通过输入缓冲器转换成合格的CMOS电平，并转换成互补的双端信号，通过公共数据线送到单元两侧位线BL和$\overline{BL}$上。若写"1"，$V_{BL} = V_{OH}$，$V_{\overline{BL}} = V_{OL}$，不论单元原来是什么状态，$\overline{BL}$通过$M_6$强迫对结点2放电，使$V_2$下降为低电平；而BL则通过$M_5$对结点1充电，使$V_1$上升为高电平。写"0"过程则相反。

图4.19是整个SRAM的方框图，存储矩阵容量为8 bit×16字，8位一个单元。它总共由128个SRAM单元构成。整个电路由一套地址总线，两套数据总线，一条读控制线，两条写控制线组成。其中的地址选择是用4-16译码器来实现的。

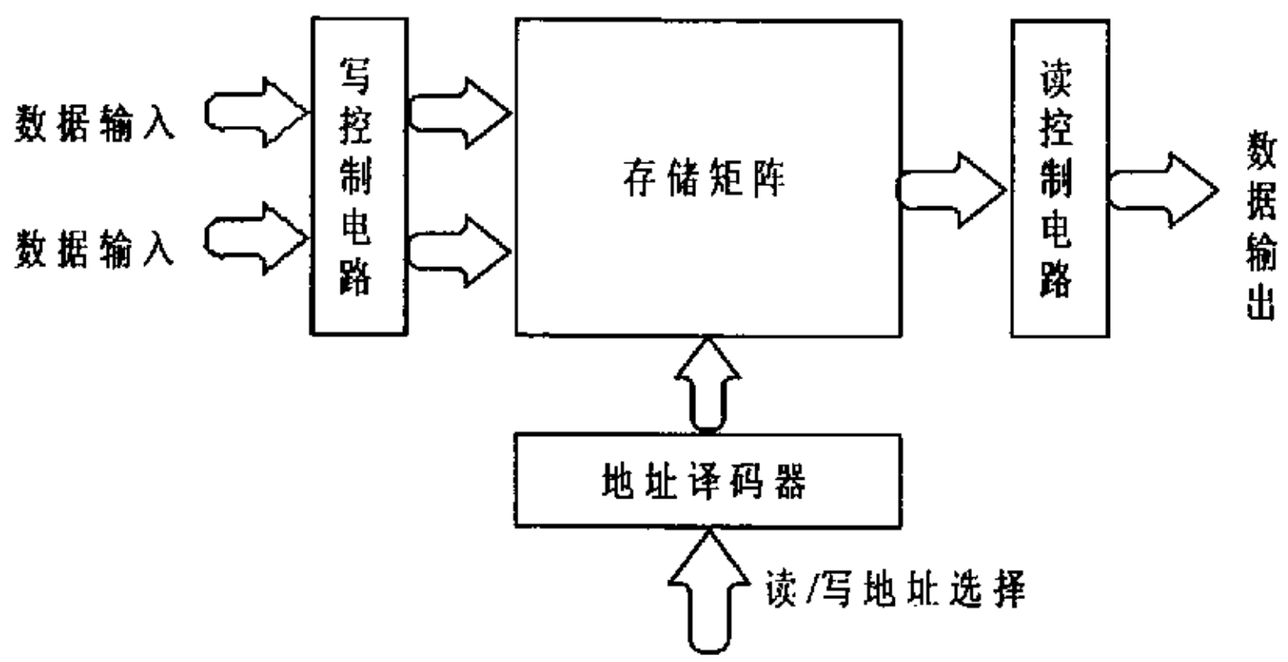


图4.19　SRAM的电路结构框图

图4.20是图4.18的符号块，RAM中的存储矩阵就是由128个这样的单元组成的。

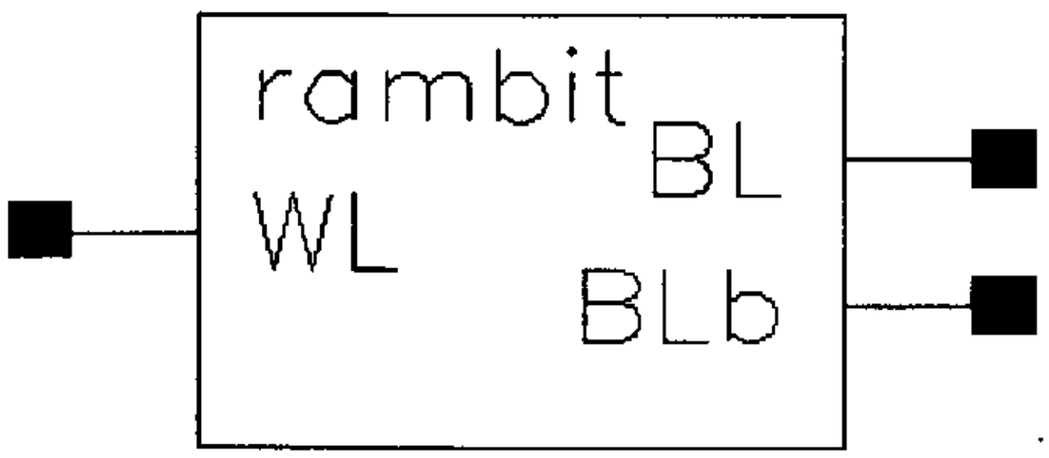


图4.20　SRAM符号块

由于每个SRAM存储单元是靠MOS管栅极对电荷的充放电来"写"和"读"数据，而Verilog-XL是事件驱动的仿真工具，针对电路图逻辑功能来工作，它可以对逻辑电路及程序文件做仿真，但不能直接对电荷存储效应做仿真，所以在对SRAM电路仿真的时候要把SRAM模块用由Verilog语言描述的模块代替掉，两模块在功能上应完全一致。

SRAM工作流程图如图4.21所示，

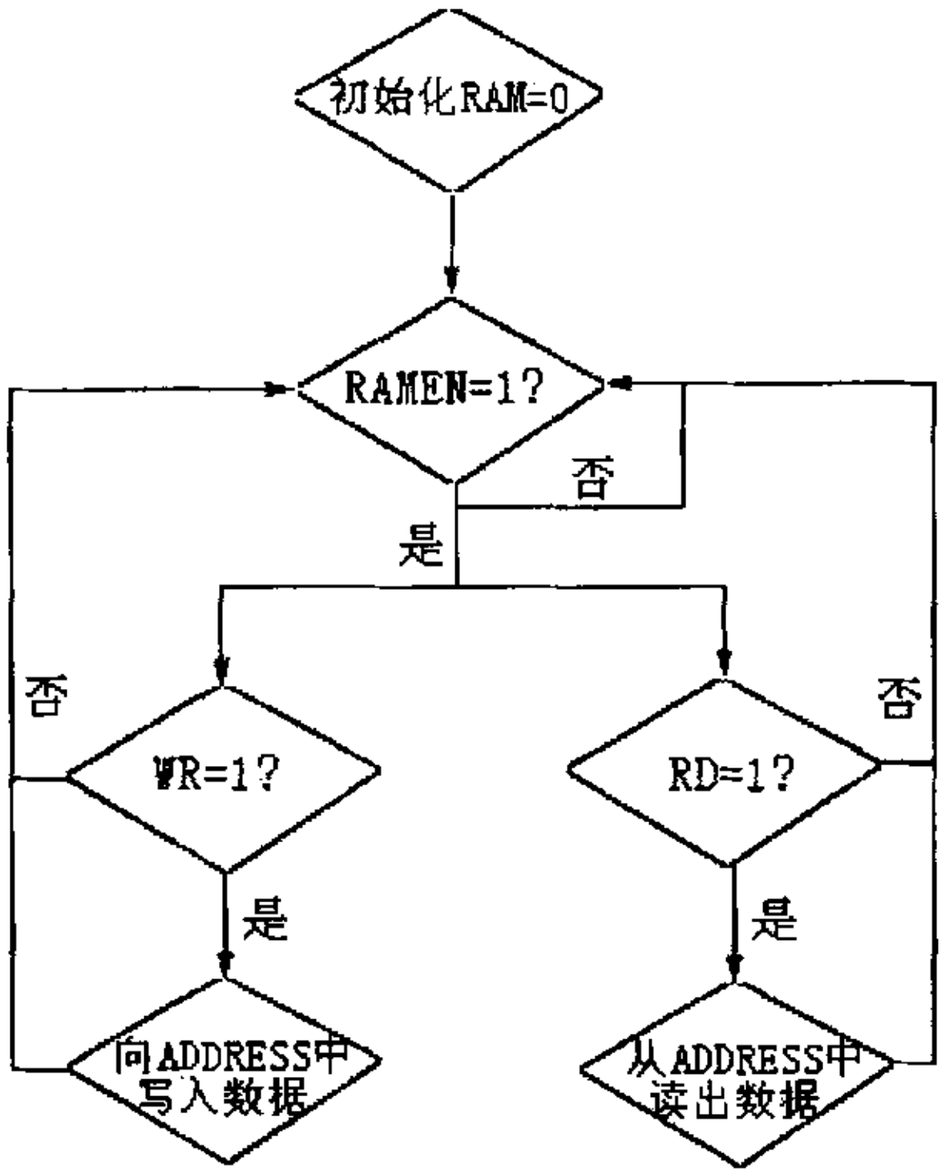


图 4.21　SRAM 工作流程图

首先，将 SRAM 初始化为全零，然后判断 RAM 使能端电平，若为“1”，则可对 SRAM 进行操作，若为“0”，则不能对 SRAM 进行读写操作，在 SRAM 工作时，分别根据写“WR”信号，读“RD”信号和地址信号，对 SRAM 读写数据。在电路实际工作中，数据实际上是以高速率集中写入，然后以低速率均匀读出，先以地址递增的方式将数据写进 SRAM，再依此将数据从 SRAM 读出。由于显示中的两个标点号经 SRAM 输出后不去 ROM 读数据而是直接经过缓冲电路输出，所以在对 SRAM 进行语言描述的时候把它分为两个 SRAM 块来进行。

源代码如下：

```
module RAM (Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, A0, A1, A2, A3, D0, D1, D2, D3,
            D4, D5, D6, D7, EN, RD, WR, WRH);
    output Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7 ;
    input A0, A1, A2, A3 ;
    input D0, D1, D2, D3, D4, D5, D6, D7;
    input EN, WR, RD, WRH ;
     reg [7:6] MEMH [0:15] ;
     reg [5:0] MEML [0:15] ; //memory matrix
     reg [7:0] Q ; //ouput latch
    wire [3:0] ADD = {A3, A2, A1, A0} ;
```

```
        assign {Y7, Y6, Y5, Y4, Y3, Y2, Y1, Y0} = Q ;

// read data from memory
  always @(posedge RD)
          if(EN == 1)
             begin
                Q[7:6] = MEMH[ADD] ;
                Q[5:0] = MEML[ADD] ;
             end
          else Q = 8'b1 ;

// write data into memory

  always @(posedge WR)
          if (EN == 1)
              MEML[ADD] = {D5, D4, D3, D2, D1, D0} ;
  always @(posedge WRH)
          if (EN == 1)
              MEMH[ADD] = {D7, D6} ;
endmodule
```

整个 RAM 模块的仿真结果如图 4.22 所示：

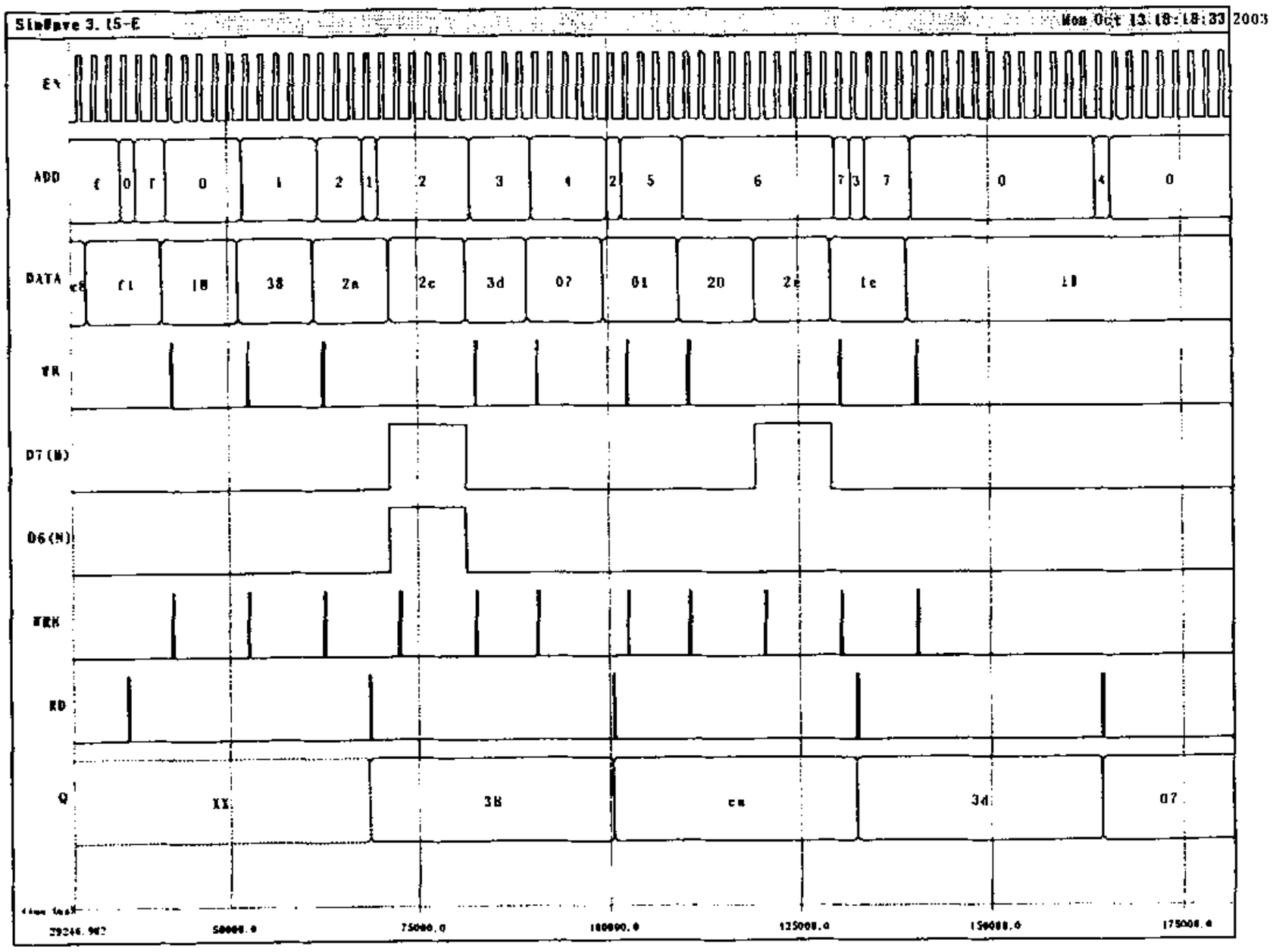


图 4.22　RAM 的 Verilog-XL 仿真结果

需要说明的是，针对两个标点符号显示的特殊性，我们不希望它们在 RAM 中也是以地址递增的方式工作，所以它们的写信号“WRH”与 RAM 写信号“WR”独立，而读信号“RD”是共用的。在命令译码模块中我们为两个标点设置的地址码是“2C”和“2E”，当输入数据为这两个码时，M 或 N 信号将产生高跳变，同时，在“WRH”信号下，数据被写入当前 RAM 地址中，并在“RD”信号控制下，随同 D6~D0 一起被读出。

4．7．2 只读存储器（ROM）电路的设计

为满足该“米”字型显示屏的显示需要，我们在 ROM 里内建了 ASCII 字符库，总共有 64 个。通过接收译码电路的地址，输出需要显示的字符内容。

ROM 的电路结构包含了存储矩阵、地址译码器和输出缓冲器三个组成部分，如图 4.23 所示。存储矩阵由许多存储单元排列而成。存储单元可以用二极管构成，也可以用双极型三极管或 MOS 管构成。每个单元能存放 1 位二值代码（0 或 1）。每一个或一组存储单元有一个对应的地址代码。

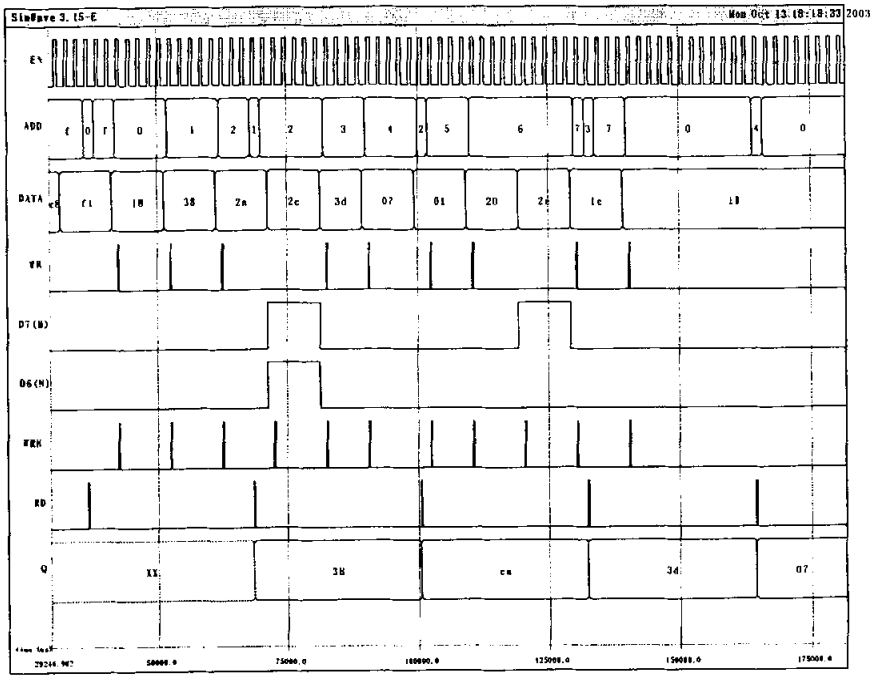


图 4.22　RAM 的 Verilog-XL 仿真结果

需要说明的是，针对两个标点符号显示的特殊性，我们不希望它们在 RAM 中也是以地址递增的方式工作，所以它们的写信号“WRH”与 RAM 写信号“WR”独立，而读信号“RD”是共用的。在命令译码模块中我们为两个标点设置的地址码是“2C”和“2E”，当输入数据为这两个码时，M 或 N 信号将产生高跳变，同时，在“WRH”信号下，数据被写入当前 RAM 地址中，并在“RD”信号控制下，随同 D6~D0 一起被读出。

4．7．2 只读存储器（ROM）电路的设计

为满足该“米”字型显示屏的显示需要，我们在 ROM 里内建了 ASCII 字符库，总共有 64 个。通过接收译码电路的地址，输出需要显示的字符内容。

ROM 的电路结构包含了存储矩阵、地址译码器和输出缓冲器三个组成部分，如图 4.23 所示。存储矩阵由许多存储单元排列而成。存储单元可以用二极管构成，也可以用双极型三极管或 MOS 管构成。每个单元能存放 1 位二值代码（0 或 1）。每一个或一组存储单元有一个对应的地址代码。

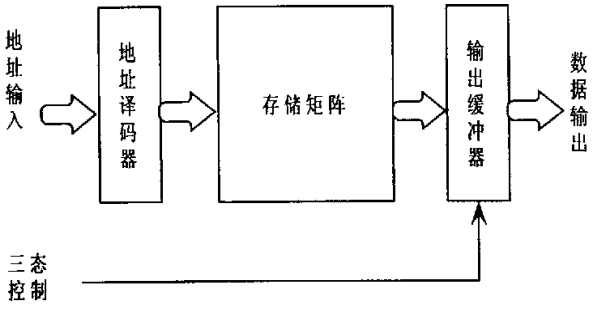


图 4.23　ROM 的电路结构框图

地址译码器的作用是将输入的地址代码译成相应的控制信号，利用这个控制信号从存储矩阵中把指定的单元选出，并把其中的数据送到输出缓冲器。

输出缓冲器的作用有两个，一是提高存储器的带负载能力，二是实现对输出状态的控制。

这里对地址译码器、存储矩阵的设计中，我们采用了预充-求值的动态逻辑电路。静态逻辑电路中靠稳定的输入信号使 MOS 晶体管保持在导通或截止状态，从而维持稳定的输出状态。输入信号存在，对应的输出状态存在；只要不断电，输出信息可以长久保持。

在动态电路中利用电容的存储效应来保存信息，因此即使输入信号不存在，输出状态也可以保持，但是信息不能长期保持，会由于泄漏电流的存在使存储的信息丢失。

CMOS 电路本身就是无比电路，静态功耗很低，在 CMOS 电路中采用动态逻辑主要是为了简化线路，减少器件，从而减小芯片面积，因为在动态 CMOS 逻辑电路中不要求 NMOS 管和 PMOS 管成对出现。另外 CMOS 动态逻辑电路有利于提高速度，这一方面是由于减小了输入电容，另一方面是由于动态逻辑电路常采用“预充-求值”的工作方式，在预充期间使输出预先充电到高电平或者放电到低电平，从而避免了上升时间或下降时间对速度的影响。

图 4.24 是我们运用的一个实际的预充-求值工作方式的动态 CMOS 与非门。

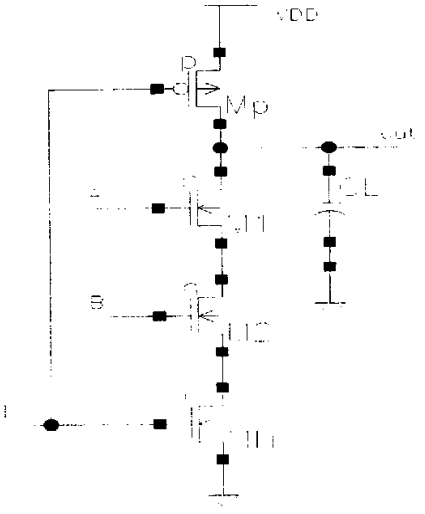


图 4.24　预充-求值的动态 CMOS 与非门

当 I= 0 时电路处于预充阶段，M_P 导通对输出结点电容充电，由于 M_N 截止，下拉通路断开，使输出电平 V_{out} 达到高电平 V_{DD}。当 I= 1 时，M_P 截止上拉通路断开，由于 M_N 导通，使下拉通路可以根据输入信号求值。若 A=B=1 则整个下拉通路导通，使输出放电到低电平；否则 M_1 和 M_2 中至少有一个管子截止，不能形成放电通路，输出保持预充的高电平。由以上分析看出，这个电路在 I= 1 时实现了 $\overline{AB}$ 的功能。这种预充-求值的动态电路克服了纹波动态电路的问题，用一对受时钟信号控制的 NMOS 管和 PMOS 管使上拉通路和下拉通路不能同时导通，因此是一种无比电路。这个电路中是用 NMOS 逻辑块实现逻辑功能的，NMOS 管占大多数，因此是富 NMOS 电路。

图 4.25 所示的是译码器及一个 ROM 单元电路。整个 ROM 存储器由 16 个类似这样的单元电路组成，分别对应于“米”字屏中 A～P16 个段。每块单元电路由 8×8 的存储矩阵构成，整个 ROM 是一个 8×128 的矩阵。译码器由行地址译码器和列地址译码器构成。输入地址的高 3 位加到行地址译码器上，从 8 行存储单元中选出要读的一行。输入地址的低 3 位加到列地址译码器上，再从选中的一行存储单元中为每个 ROM 单元选出要读的一位，这时，ROM 中的数据便通过 A～P 口出现在输出端上。译码器的充放电过程由“EN”信号和“PLA_RD”信号控制，当 EN = 1 或 PLA_RD = 1 时，PMOS 管导通，NMOS 管截止，实现预充电过程，事实上，此时的电荷存储在 PMOS 管的栅电容上，当 EN = 0 或 PLA_RD = 0 时，PMOS 管截止，NMOS 管导通，输出结点通过行列译码器选出的一位放电，如果位线与字线的交叉点上有 NMOS 管，则放电成功，输出为低电平，如果没有，则放电失败，输出为高电平。

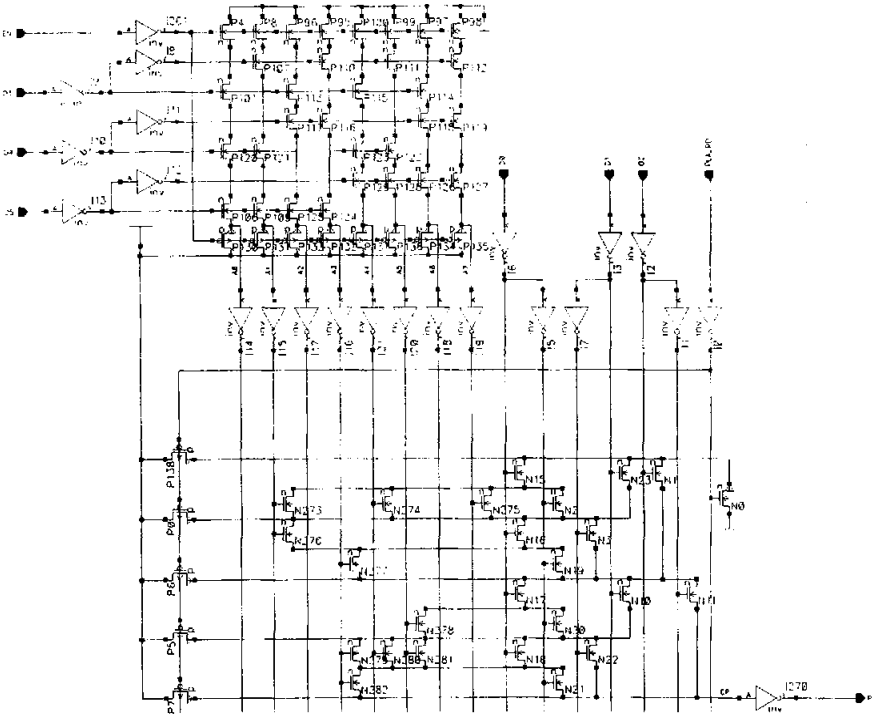


图 4.25　ROM 单元及译码器电路结构

和 RAM 同样的道理，在对 ROM 进行仿真时，需要用语言描述来替换电路，两者在行为功能上完全一致。

源代码如下：

```
module PLA ( A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, D0, D1,
        D2, D3, D4, D5, EN, PLA_RD );

output   A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P;
input   D0, D1, D2, D3, D4, D5, EN, PLA_RD;
trireg A0, A1, A2, A3, A4, A5, A6, A7;
trireg CA, CB, CC, CD, CE, CF, CG, CH, CI, CJ, CK, CL, CM, CN, CO, CP;
supply0 gnd ;
supply1 vdd ;

nmos P98( net4123, gnd, net3785);
.....
tranif1 N200( net5880, CA, net4724);
.....
pmos P138( net3861, vdd, net1060);
```

```
.....
not I261 ( net3785, EN);
.....
endmodule
```

源代码是通过 Verilog 门级和开关级描述来实现的，为了反映栅电容的电荷储存效应，将连接到栅的连线说明为具有电荷保持作用的 trireg 型，其中省略号表示的都是结点间的连接关系，由于比较繁琐，故略去。

仿真结果如图 4.26 所示：

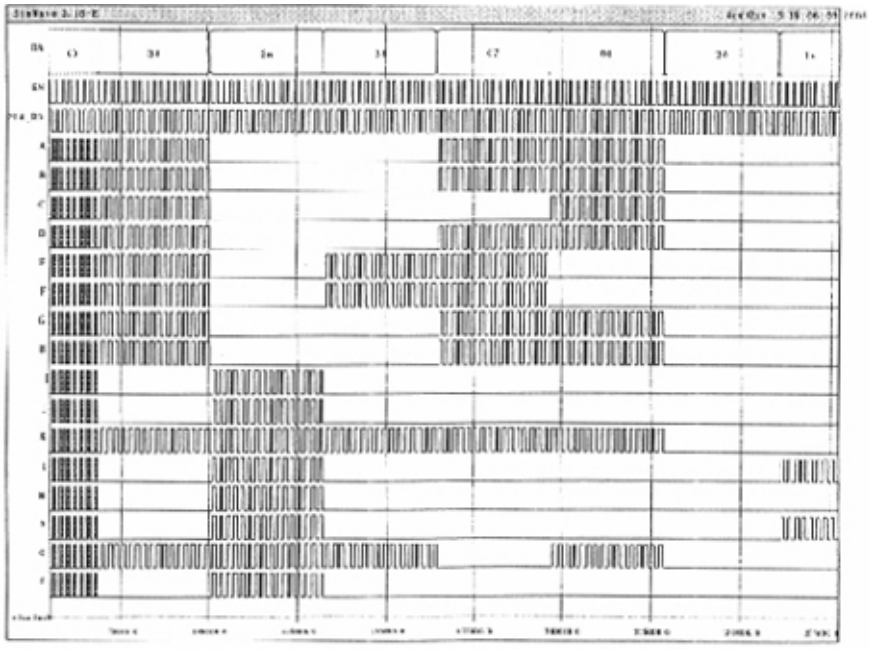


图 4.26　ROM 的 Verilog-XL 仿真结果

4．8 栅扫描电路设计

这部分电路的设计比较简单，其实就是一个顺序脉冲发生器，产生阶梯波，主要由四个锁存器和一个 4-16 译码器组成。4 位地址也就是逻辑控制模块中的读 RAM 地址，在每个 GRID 被选中期间，SEGMENT 随显示存储器的内容而变化，这样便实现了输出数据与栅扫描信号间的一一对应关系。脉宽则由命令译码模块中的 AD_LA 和 AD_DUTY 控制。电路如图 4.27 所示：

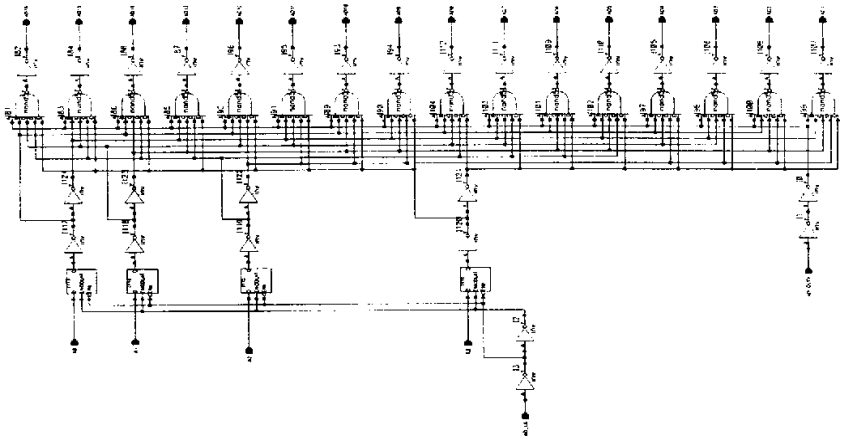


图 4.27　栅扫描电路结构

仿真结果如图 4.28 所示：

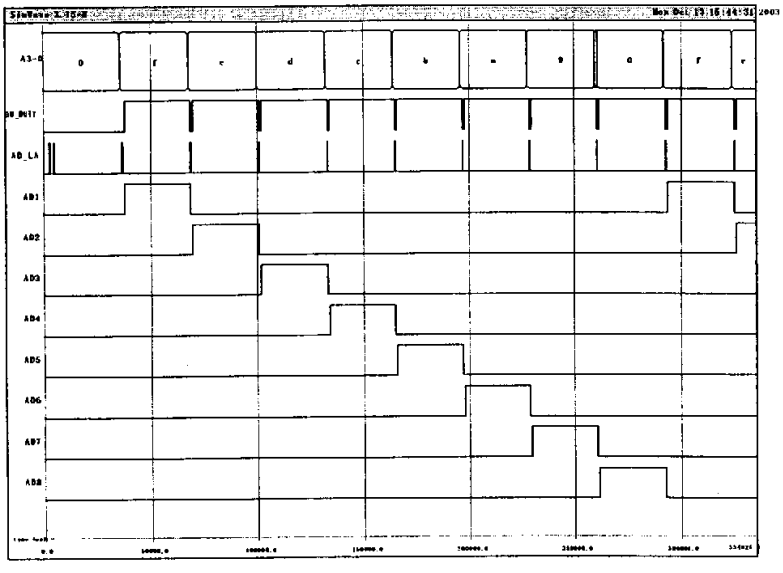


图 4.28　栅扫描电路 Verilog-XL 仿真结果

4．9 输出接口电路设计

这部分电路的主要功能是实现输出数据的保持和缓冲，主要由一系列锁存器、与非门、反相器组成。通过 SEG_LA 控制输入数据的存储，再通过 SEG_DUTY 控制数据的输出与否。

4．10 总体逻辑电路模块划分及功能仿真

4．10．1 总体逻辑电路模块划分

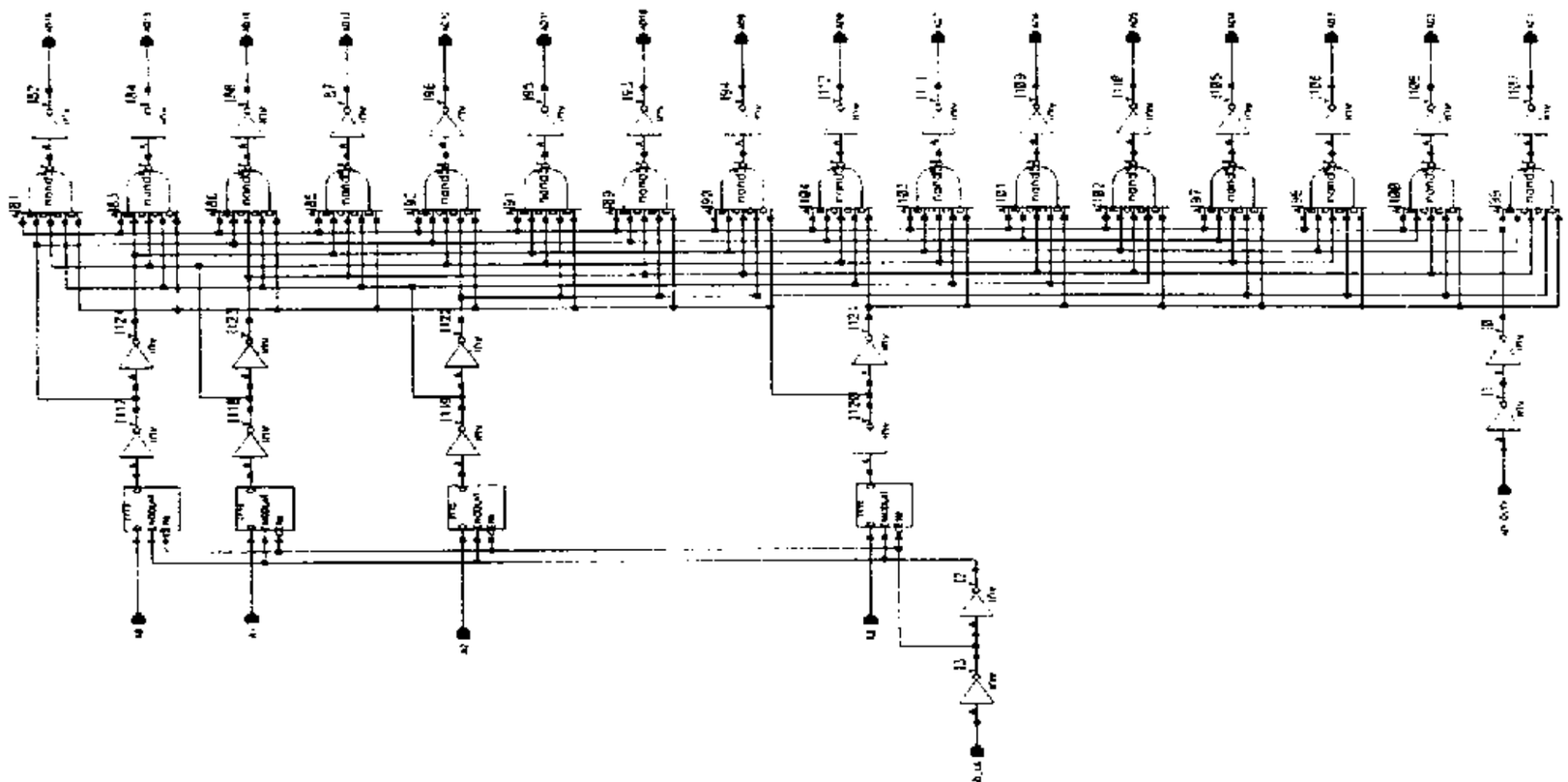


图 4.27　栅扫描电路结构

仿真结果如图 4.28 所示：

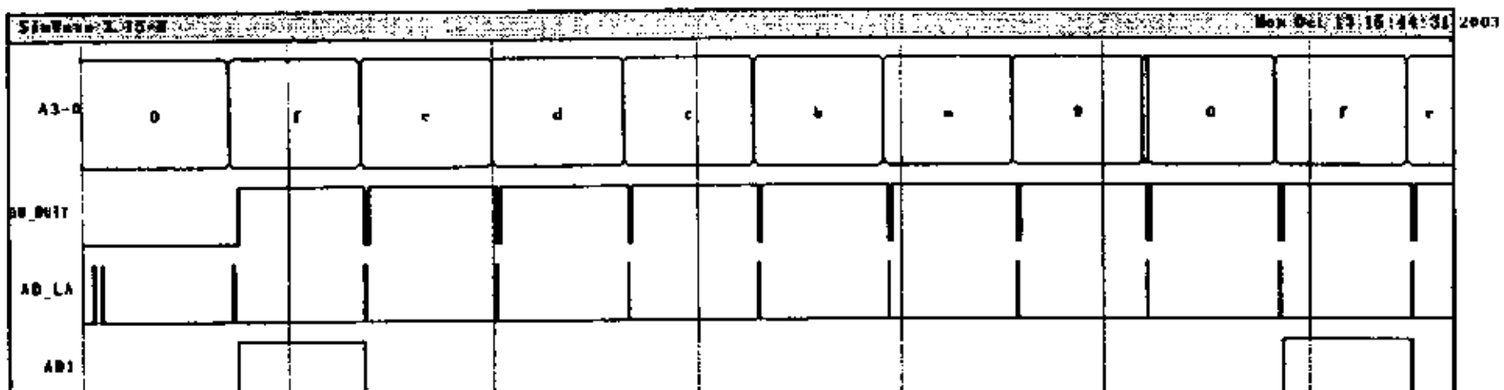


图 4.28　栅扫描电路 Verilog-XL 仿真结果

4．9 输出接口电路设计

这部分电路的主要功能是实现输出数据的保持和缓冲，主要由一系列锁存器、与非门、反相器组成。通过 SEG_LA 控制输入数据的存储，再通过 SEG_DUTY 控制数据的输出与否。

4．10 总体逻辑电路模块划分及功能仿真

4．10．1 总体逻辑电路模块划分

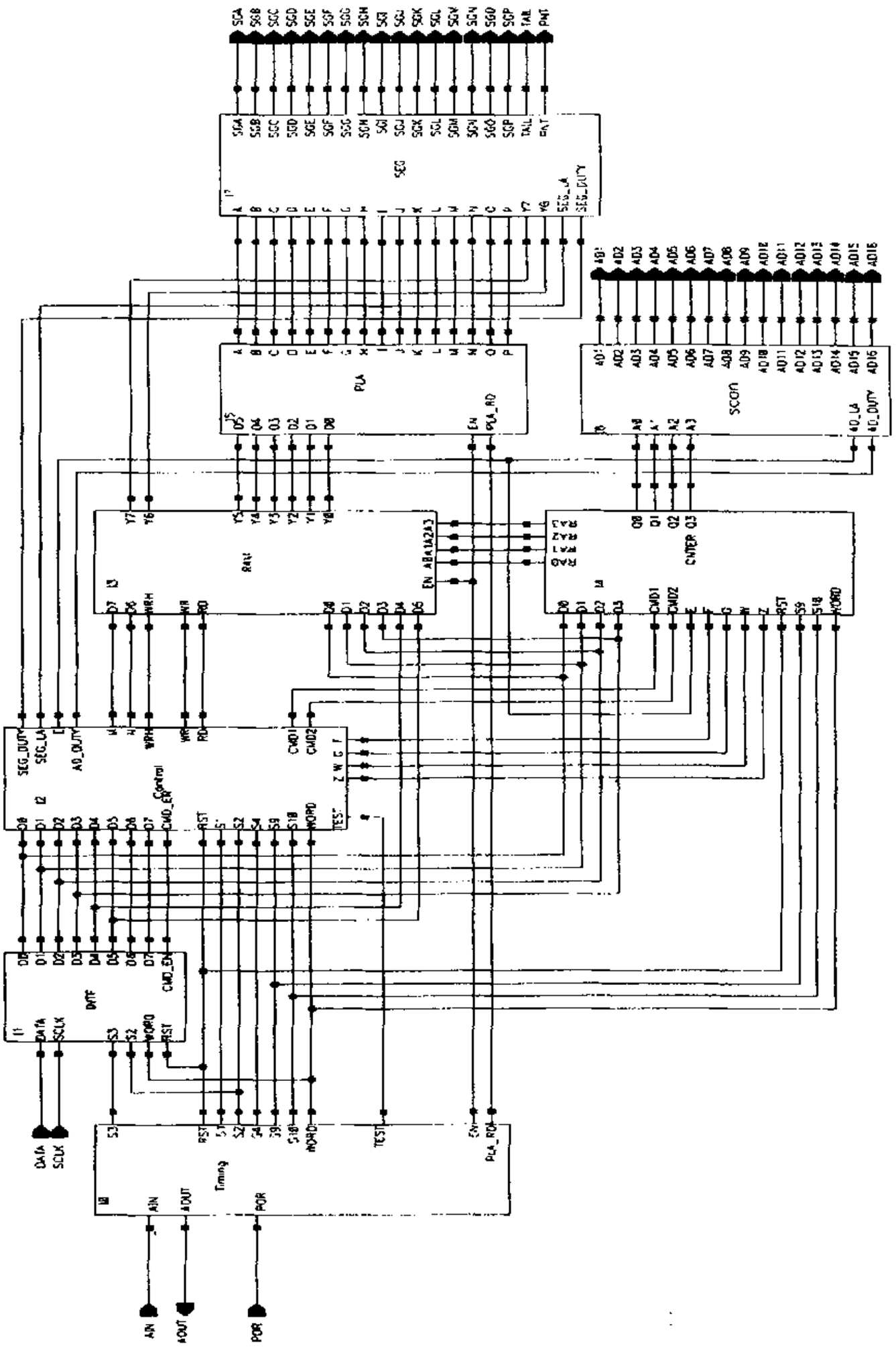
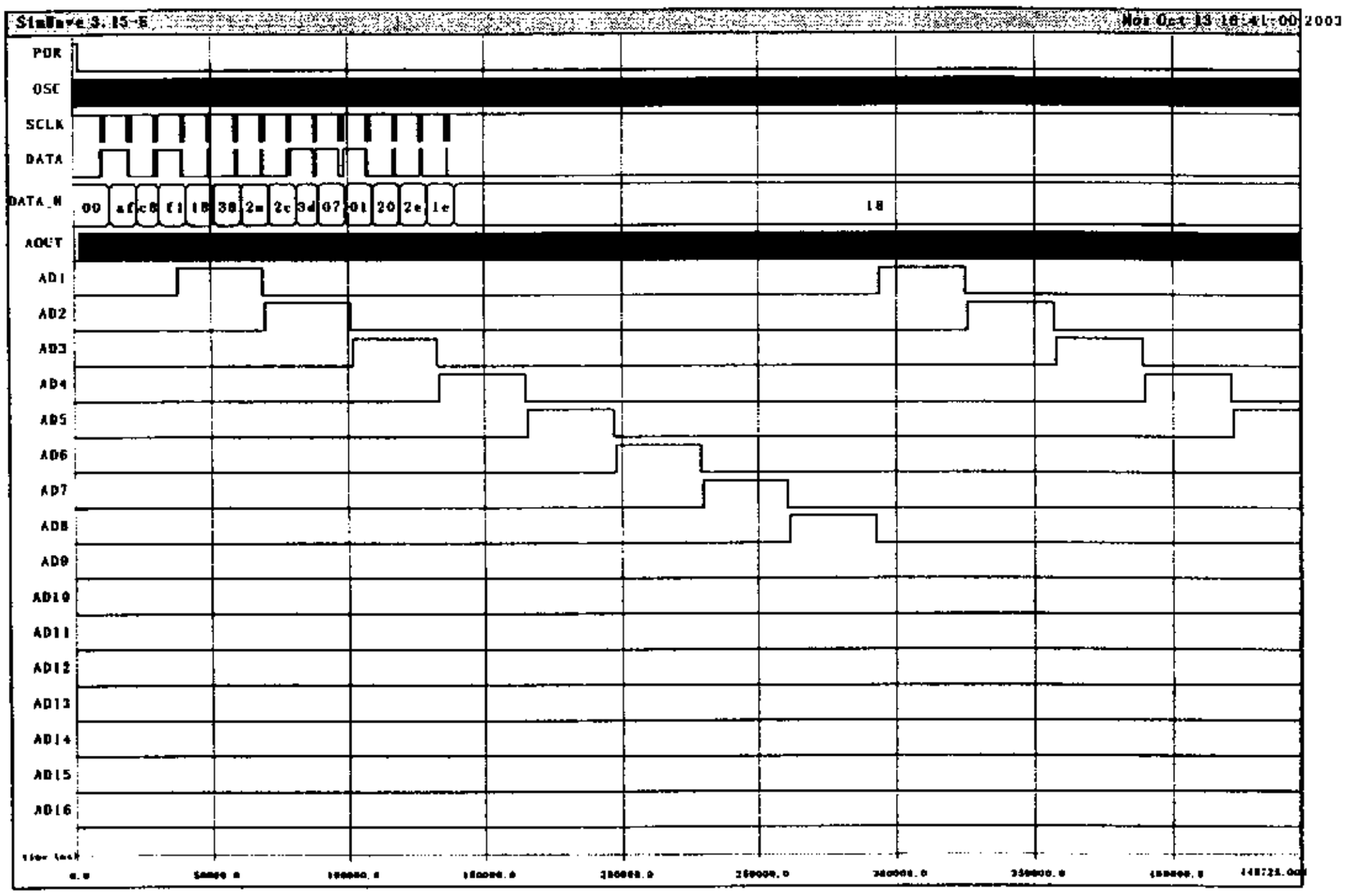


图 4.29　总体逻辑电路模块划分

4．10．2 总体逻辑电路混合级仿真结果



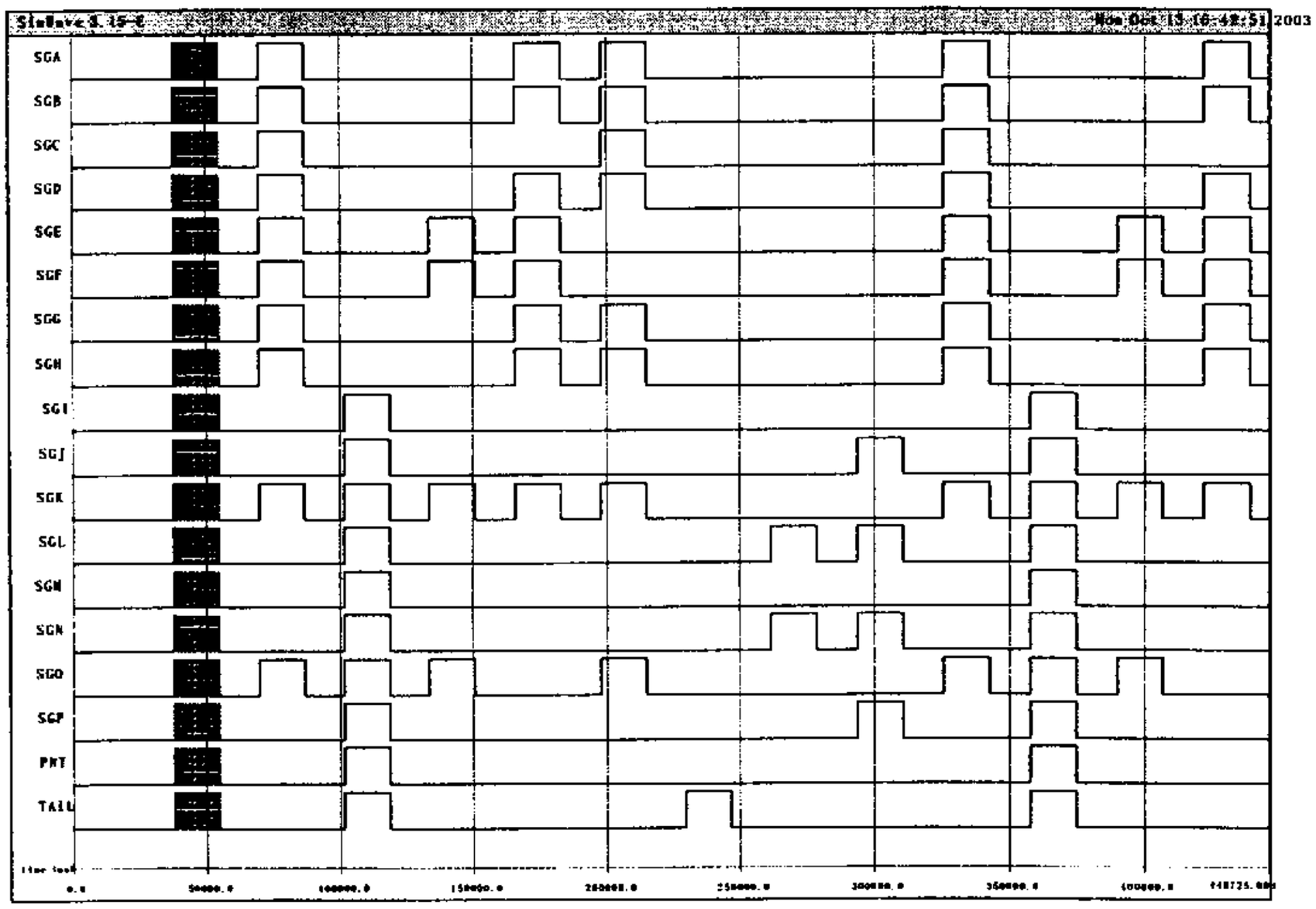


图 4.30　芯片整体功能 Verilog-XL 仿真结果

仿真测试文件见附录。

指令 / 数据	代码	功能
地址设置指令（CMD1）	AFH	从首位即 AD1 开始亮起
显示模式设置指令（CMD2）	C8H	亮 8 位即 AD1~AD8
显示亮度设置指令（CMD3）	F1H	亮度调为 15/32
数据 1	18H	见图 3.8

数据 2	38H	见图 3.8
数据 3	2AH	见图 3.8
数据 4	2CH	见图 3.8
数据 5	3DH	见图 3.8
数据 6	07H	见图 3.8
数据 7	01H	见图 3.8
数据 8	20H	见图 3.8
数据 9	2EH	见图 3.8
数据 10	1EH	见图 3.8
数据 11	18H	见图 3.8

表 4.4　仿真测试数据表

4．11 本章小结

本章根据“自顶向下”的设计思想，从整个芯片的功能要求出发，将芯片进行层次化功能划分，同时，参考已有的同类驱动芯片的设计经验，结合“自底向上”的原理图输入设计方法对各模块进行具体的电路设计或描述，并进一步由底层次模块构成上一层次模块，直至构成最高层，即实现整个芯片的功能，最后根据系统设计的框架对各模块进行协调设计，并进行芯片的整体功能验证，从而达到设计的目的。在芯片的功能验证中，我们采用了 Verilog 硬件描述语言进行测试文件描述，对电路的逻辑功能和时序关系进行了仿真验证。

第五章　VFD 驱动芯片的版图设计与分析

芯片版图的设计我们采用了 0.6um 的硅栅 CMOS 工艺，设计规则采用首钢 NEC 的 CZ5L 工艺规则。CZ5L 是一种经过优化的三层金属线，双层多晶硅，0.6um 的 CMOS 工艺。

5．1 标准单元设计方法

我们采用了标准单元设计模式。标准单元法的特点是各个单元具有同一高度（指版图尺寸），但宽度不等。单元本身经过精心设计，并完成了设计规则检查和电学性能验证。设计好的各单元存入设计系统的物理单元库中以便调用，单元的逻辑符号及电学特性则存入逻辑库中。

标准单元法的版图结构见图 5.1：

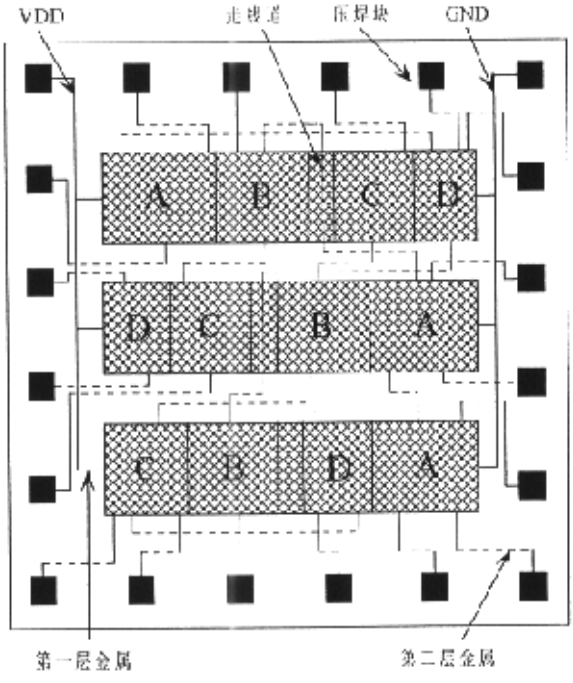


图 5.1　标准单元版图结构

设计时将所需单元从单元库中调出，将其排列成若干行，行间留有布线通道。然后根据电路要求将各单元用连线连接起来，同时把相应的输入/输出单元和压焊块连接起来，就得到所要求的芯片版图。

芯片主要分为 3 个区域：1）四周的 I/O 单元和压焊块；2）单元部分；3）布线通道。由于标准单元本身的信号端都引到单元的上下两端，因此单元之间的连线都处在布线通道内。

标准单元法的布局和布线都是自动进行的。我们只要输入被设计电路的逻辑图或输入一种电路描述文件，再输入压焊块的排列次序，标准单元法自动设计系统将调用所需的单元和相应 I/O 单元及压焊块，进行自动布局和布线。在布局和布线过程中，布线通道的高度由设计系统根据需要加以调整，当布线发生困难时，将通道间距适当加大，因而布局布线是在一种不太受约束的条件下进行的，可以保证 100%的布线布通率。

5．2 自动布局布线流程

Cadence 中进行布局规划的工具是 Preview，针对标准单元法的自动布局布线工具为 Silicon Ensemble。自动布局布线的流程主要由以下几个步骤组成：

1）准备好自动布局布线库

在进行自动布局布线之前，必须准备好相应的库。该库中含有工艺数据、自动布局布线用的库单元及显示信息。库的格式必须为 Design Framework II 的数据库格式，可以由用户利用版图生成工具 Vituoso Layout Editor 设计产生，也可以来自由芯片制造厂家和 EDA 公司提供的 LEF（Library Exchange Format）文件，或者从 GDSII 生成。

2）准备用来进行自动布局布线的网表

用来进行自动布局布线的网表可以由硬件描述语言经过综合优化或由电路提取而来。所有网表在进行自动布局布线前都必须首先生成对应的 autoLayout 视图（view）

3）用 Preview 进行布局规划

Preview 是 Cadence 的布局规划器。它可以用来规划物理设计，从而在自动布局布线前预估物理实现的影响。在 Cadence 中，使用 Preview 与自动布局布线引擎相结合来进行自动布局布线。

4）用 Silicon Ensemble 进行自动布局布线。

5）对完成布局布线的版图进行验证

生成的版图其连接性是否正确，是否符合设计规则，是否符合时序要求等等，必须通过验证才能确定。通过点击 Verify&Report 菜单中的相应项，可对版图进行连接性、设计规则验证，并可生成 SDF（Standard Delay Format）文件。通过反标 SDF 文件可对原来的门级网表进行仿真，从而确定其功能和时序是否正确。

我们只要指定了由厂家提供的 CZ5L.tf 工艺库文件、标准单元库文件及从逻辑电路中提取的网表文件，就可以通过Preview和Silicon Ensemble工具完成版图的自动布局布线。

5．3 版图设计规则及验证

5．3．1 版图设计规则

图5.2所示的是一个MOS晶体管的结构示意图：

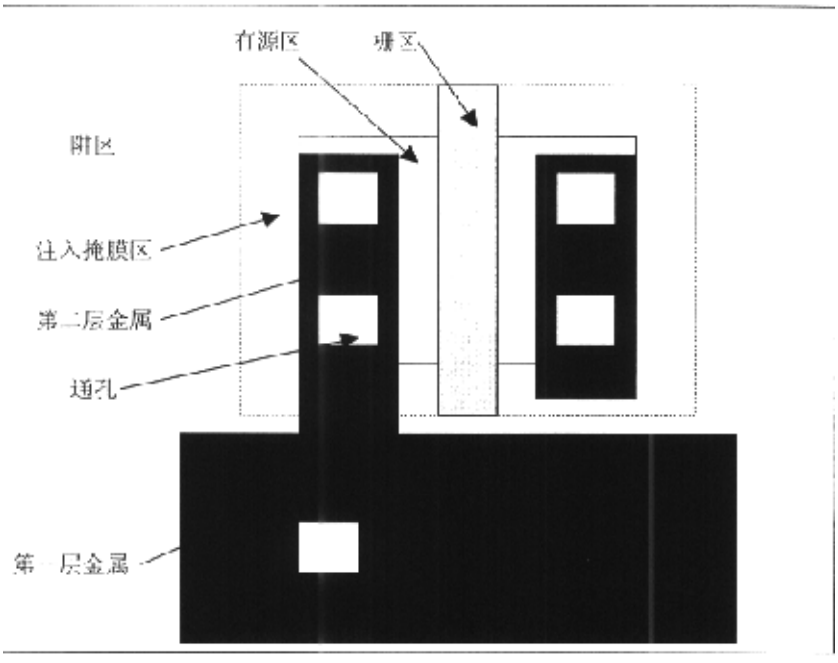


图 5.2 MOS 晶体管的结构示意图

版图绘制要根据一定的设计规则来进行，也就是说一定要通过DRC（Design Rule Checker）检查。首钢NEC CZ5L工艺的部分设计规则如下：

1.a	n 阱(well)	n 阱的最小宽度	1.60um
1.b		阱与阱之间的最小间距	1.60um
1.c		ndiff 到 nwell 的最小间距	0.80um
1.d		pdiff 到 nwell 的最小间距	0.80um
1.e		pmos 器件必须在 nwell 内	
2.a	有源区（active）	有源区的最小宽度	1.00um
2.b		有源区之间的最小间距	0.64um
3.a	多晶硅（poly）	多晶硅的最小宽度	0.64um
3.b		多晶硅间的最小宽度	0.64um
3.c		多晶硅与有源区的最小间距	0.08um
3.d		多晶硅栅在场区上的最小露头	0.48um
3.e		源、漏与栅的最小间距	0.60um
4.a	引线孔（contact）	引线孔的最小宽度	0.64um
4.b		引线孔间的最小间距	0.64um
4.c		多晶硅覆盖引线孔的最小间距	0.40um
4.d		扩散区覆盖引线孔的最小间距	0.32um
5.a	金属 1（metal1）	金属 1 的最小宽度	0.88um

5.b		金属 1 间的最小间距	0.72um
		金属 1 覆盖引线孔的最小间距	0.28um
6.a	金属 2（metal2）	金属 2 的最小宽度	1.12um
6.b		金属 2 间的最小间距	0.72um
6.c		金属 2 覆盖通孔的最小间距	0.36um
7.a	通孔 1（via1）	通孔 1 的最小宽度	0.80um
7.b		通孔 1 间的最小间距	0.88um
7.c		通孔 1 与引线孔间的最小间距	0.60um
7.d		metal1 覆盖通孔 1 的最小间距	0.32um
8.a	通孔 2（via2）	通孔 2 的最小宽度	0.80um
8.b		通孔 2 间的最小间距	0.88um
8.c		通孔 2 与引线孔间的最小间距	0.80um
8.d		金属 2 覆盖通孔 2 的最小间距	0.36um
9.a	金属 3（metal3）	金属 3 的最小宽度	1.12um
9.b		金属 3 间的最小间距	0.72um
9.c		金属 3 覆盖通孔 2 的最小间距	0.36um

结合上述规则，我们就可以编写出相应的 DRC 规则检查文件。

编辑好的版图通过了设计规则的检查后，有可能还有错误，这些错误不是由于违反了设计规则，而是可能与实际线路图不一致造成。版图中少连了一根铝线这样的小毛病对整个芯片来说都是致命的，所以编辑好的版图还要通过LVS（Layout Versus Schematic）验证。同时，编辑好的版图通过寄生参数提取程序来提取出电路的寄生参数，电路仿真程序可以调用这个数据来进行后模拟。

5．3．2 版图验证

Cadence的版图设计工具是Vituoso Layout Editor，即为版图编辑大师。版图设计得好坏，其功能是否正确，必须通过验证才能确定。Cadence 中进行版图验证的工具主要有Dracula和Diva。两者的主要区别是，Diva 是在线的验证工具，被集成在Design Frame Work II 中，可直接点击版图大师上的菜单来启动。而Dracula 是一个单独的验证工具，可以独立运行。

我们采用的是 Cadence 环境下集成的验证工具集 DIVA。下面先对 DIVA 作一个简单介绍。

DIVA 是 Cadence 软件中的验证工具集，用它可以找出并纠正设计中的错误：它除了可以处理物理版图和准备好的电气数据，从而进行版图和线路图的对查（LVS）外。还可以在设计的初期就进行版图检查，尽早发现错误并互动地把错误显示出来，有利于及时发现错误所在，易于纠正。

DIVA 工具集包括以下部分：

1）设计规则检查（iDRC）

2）版图寄生参数提取（iLPE）

3）寄生电阻提取（iPRE）

4）电气规则检查（iERC）

5）版图与线路图比较程序（iLVS）

需要提到的是：Diva中各个组件之间是互相联系的，有时候一个组件的执行要依赖另一个组件先执行。例如：要执行LVS就先要执行DRC。在Cadence系统中，Diva集成在版图编辑程序Virtuoso和线路图编辑程序Composer中，在这两各环境中都可以激活Diva。要运行Diva前，还要准备好规则验证的文件。这些文件有各自的默认名称，如：做DRC时的文件应以divaDRC.rul命名，做ERC时的文件应以divaERC.rul命名，版图提取文件以divaEXT.rul命名，做LVS时规则文件应以divaLVS.rul命名。

通过DRC验证的版图还需要进行LVS，也就是版图和线路图对查比较。实际上就是从版图中提取出电路的网表来，再与线路图的网表比较。在diva中，由于版图提取在extract中就已经完成，LVS文件中的逻辑结构相对就比较简单。只需进行网表比较，参数比较，以及把一些“并联或串联”的元器件归并等即可。所以这一部分文件不会因为工艺层次不同而有很大不同，可以根据范本做少许改动。

5．4 版图的实现

经过了标准单元设计模式下的自动布局布线、版图验证后我们得到了如下图5.3所示的实际版图：

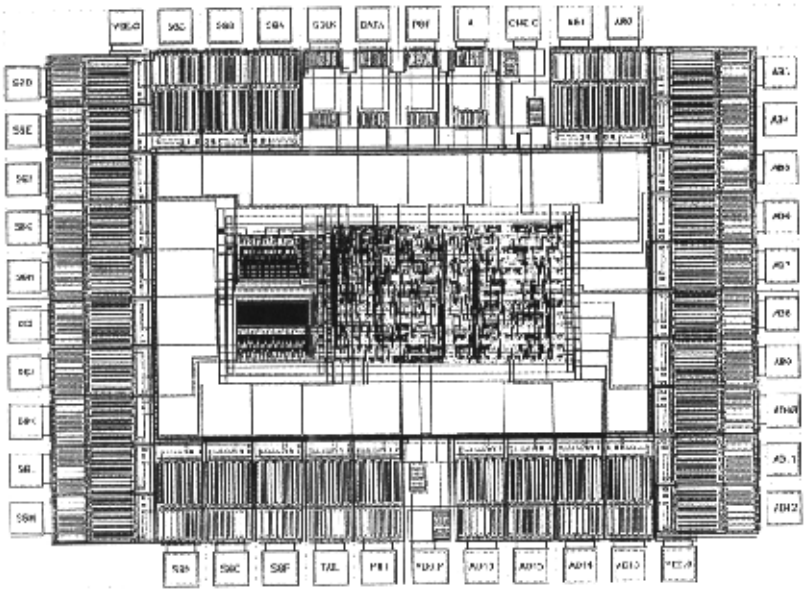


图5.3　最终设计完成的芯片版图

由图可以看出，实际的版图采用了与全定制设计法相结合的方法，即插入了宏单元（RAM、ROM）。电源线、地线、信号线的位置都是规范化的。电源线和地线从单元的左右边进出，所有的信号端从上部和下部进出。电源线和地线处在单元的最上端和最下端，信号端口在单元的上下边界处。

5．5 输入、输出缓冲器

一个集成电路芯片是通过压点和外界联系的，接收片外的输入信号，并产生输出信号驱动片外的负载。需要接到压点的输入、输出端，一般都经过输入、输出缓冲器和外界相连，使片内信号和片外信号匹配。

5．5．1 输入电路逻辑端口结构

在MOS集成电路中，正常的工作电压不会引起栅氧化层击穿。但是，作为和外界相通的MOS晶体管的栅极，即集成电路的输入端，会受到外界的各种干扰而形成很高的栅压。由于MOS晶体管栅极和其他电极之间是绝缘的，外界引入的各种杂散电荷将在栅上积累，形成等效栅压，这种静电引起的等效栅压将会造成栅击穿。

为了防止MOS IC中接到芯片输入端的MOS晶体管出现栅击穿，必须在MOS IC的输入端增加保护电路，用来为栅上积累的静电荷提供放电通路，保护连接输入压点的MOS管的栅。输入电路逻辑端口结构如图5.4所示：

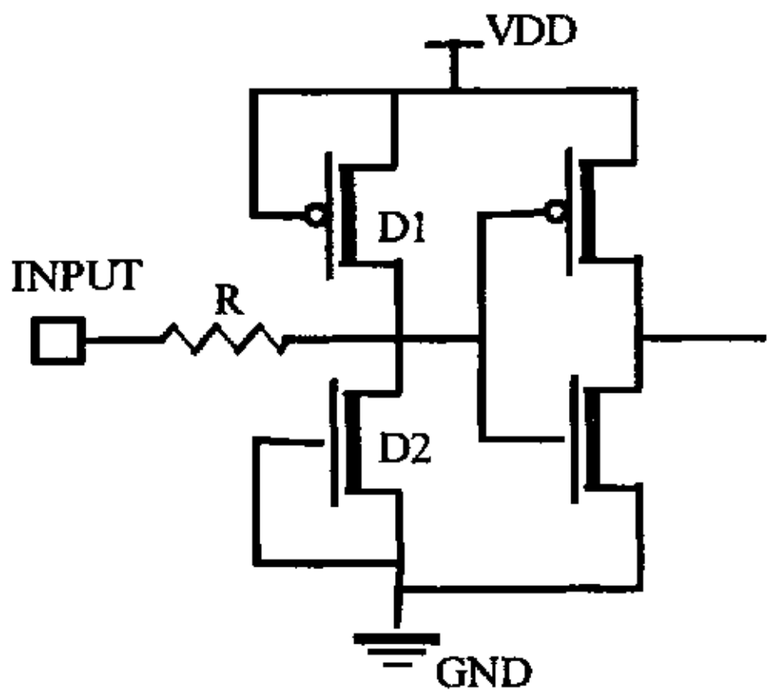


图5.4　输入电路逻辑端口结构

图中R为多晶硅电阻，D1、D2为等效于二极管的MOS管。当外界干扰或静电感应使输入端有很高的电压时，高电压可以使二级管击穿。所以只要设计二极管的击穿电压小于MOS晶体管的栅击穿电压，则外界引起的高电压，首先使二极管击穿，产生的大电流在R上引起压降，从而使加在MOS晶体管栅极的电压降低，防止了栅击穿。电阻R还有限流的作用，防止二级管击穿引起过大的电流而被烧坏。由于干扰信号包括静电引起的输入端的高电压都是瞬时的脉冲信号，只要电流不是非常大，二级管不会被烧坏，从而可以继续起保护作用。

5．5．2 输出电路逻辑端口结构

同样的道理，输出电路逻辑端口结构如图5.5所示：

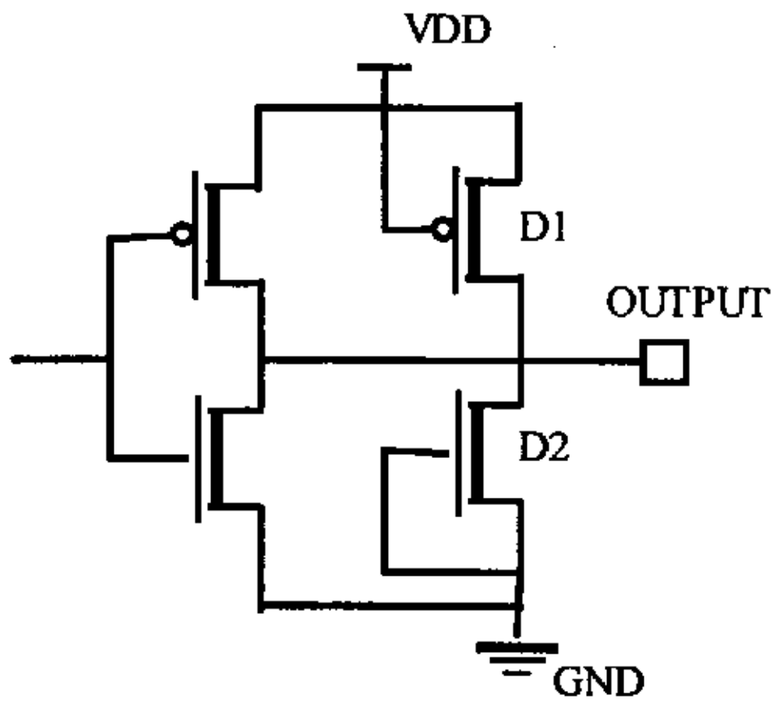


图5.5　输出电路逻辑端口结构

5．5．3 输出驱动电路结构

芯片版图中除了逻辑电路部分外，高压输出驱动电路部分和输出掩膜电阻部分也是芯片的重要组成部分，这部分版图采用了耐高压DMOS工艺。

输出驱动电路结构如图5.6所示：

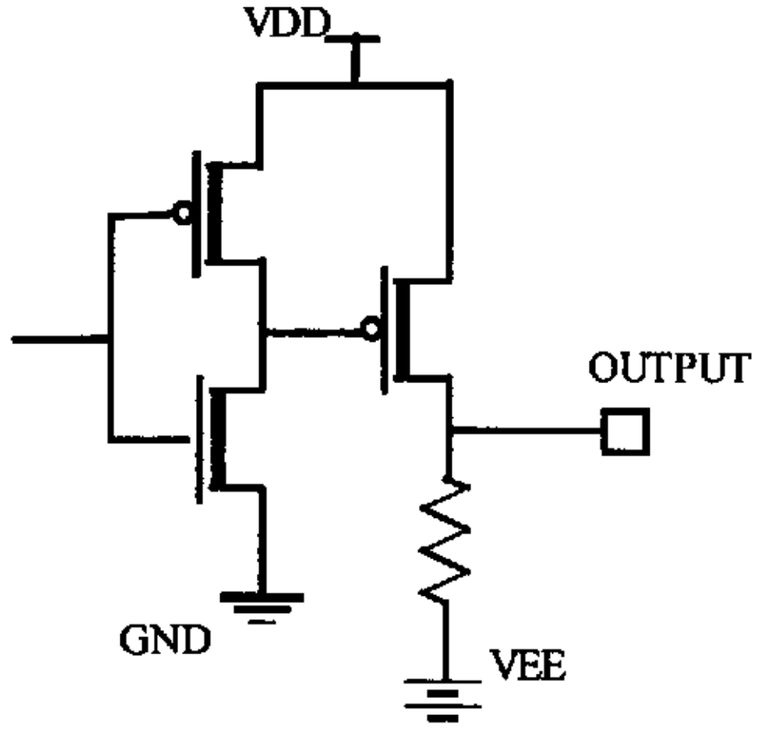


图5.6　输出驱动电路结构

这其实是一个电平转换的接口电路。输入为CMOS电平，高压输出可达到-58V，它为负载提供了高电压、大电流的输出，其高压输出直接驱动VFD屏的显示单元。

而掩膜进芯片内部的下拉电阻，当输出为高电平时，能向外提供下拉电流负载，同时，在关闭对显示屏的扫描脉冲信号时，可以加快振荡衰减。在已施加电压（ON）的状态下，下拉电阻则是一种与显示屏并联的负载，会消耗无效电力，因此取值不能太小，反之，如果取值太大，容易产生振荡，或因尖峰电压（SPIKE NOISE）而有漏光的现象，通常以数10KΩ（欧姆）左右最为合适。把外部电阻掩膜进芯片内部在同类VFD专用驱动芯片中是具有创

新意义的，避免了繁琐的外围电阻电路。

低压和高压MOS管集成在同一芯片上，使其在工艺上兼容成为设计的关键之一。低压部分MOS管的设计规则前面已经提过，高压部分MOS管的设计规则采用的首钢NEC CZ5L的高压设计规则。高压晶体管的设计规则如下图所示：

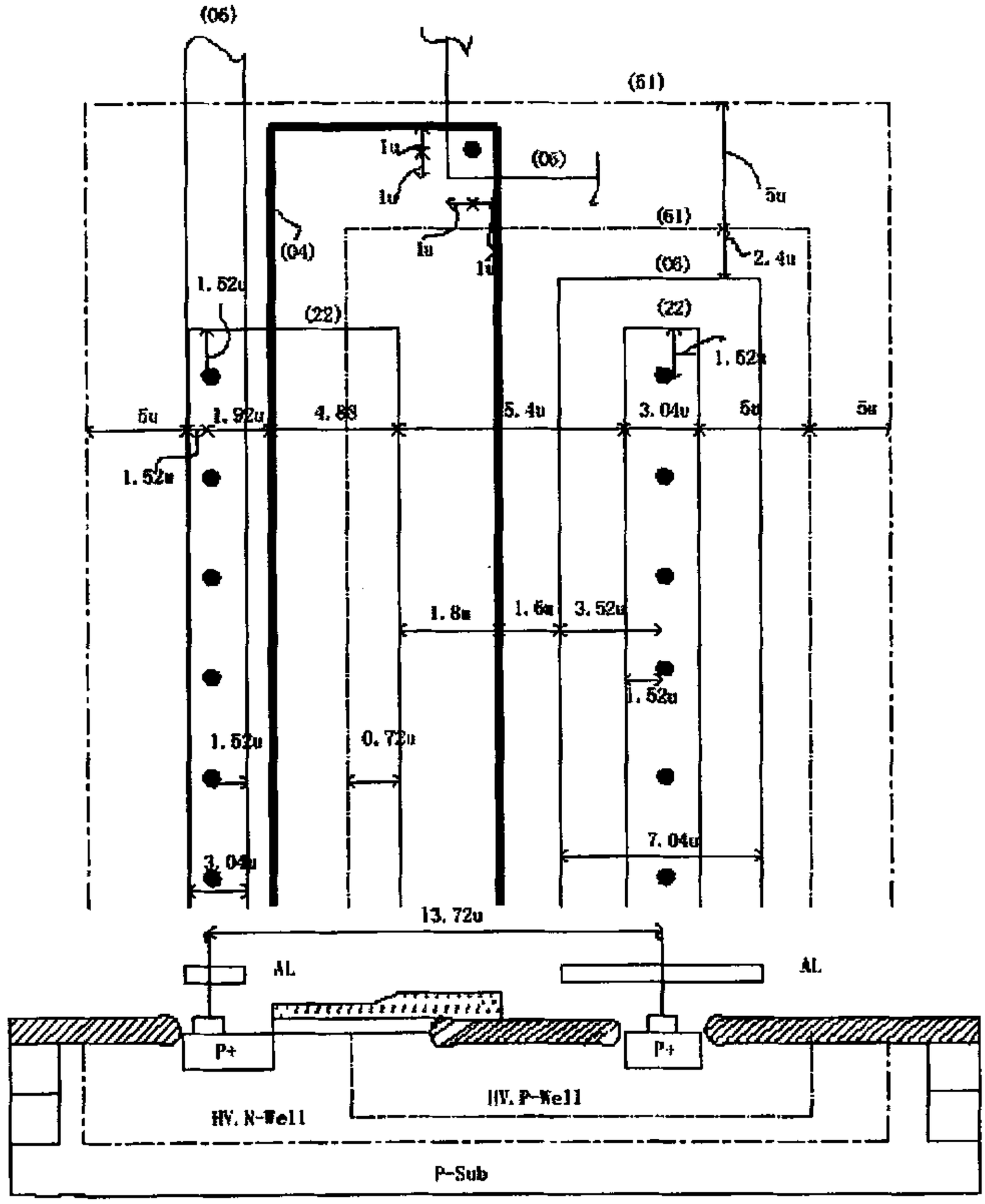


图5.7　高压晶体管设计规则

高压电阻的设计规则如下图所示：

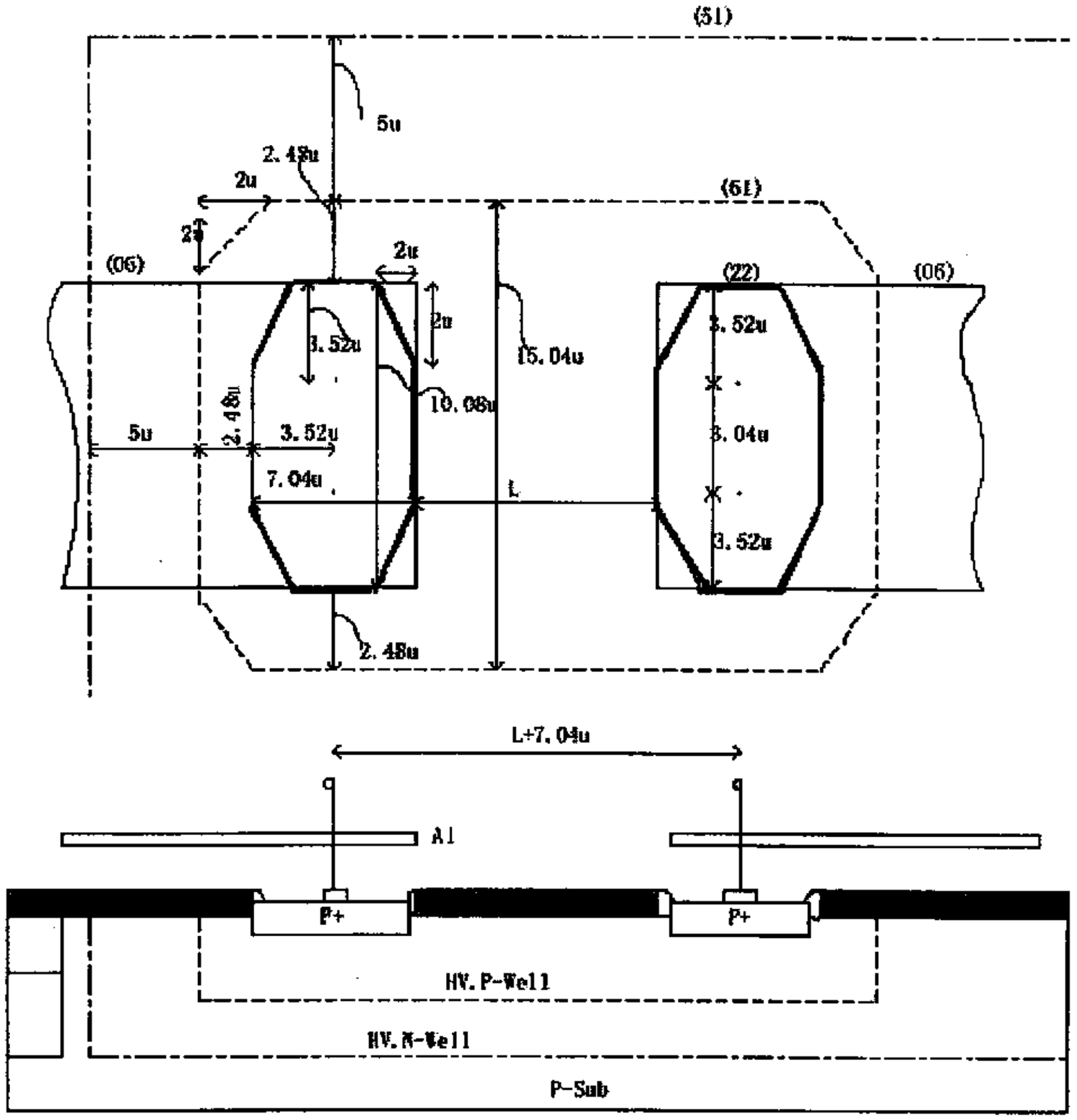


图 5.8　高压电阻设计规则

在这里，电阻值可以通过公式计算，L（um）=2.5*0.32*R（KΩ）（±10%）

5．6 掩膜生成

掩膜又称光刻版。掩膜图形从版图获得，通常有以下几条途径：

（1）直接对应。一块掩膜直接与版图中某图形层对应。

（2）逻辑组合。一块掩膜与版图中某几层图形有关，并为它们的某种逻辑运算的结果。如 P^+ 注入与 N^+ 注入正好互补，版图中可只画 P^+ 注入层（PPLUS）图形，直接对应 P^+ 注入的掩膜，N^+ 注入的掩膜便用版图中 PPLUS 层图形的“非”（NOT BULK PPLUS）得到。

（3）周边拓展。一块掩膜上的图形由版图中某种图形拓展而来。如某 MOS 器件的栅注入框，在版图中可以不画，而栅注入的掩膜层可由栅区图形拓展一定量得到。

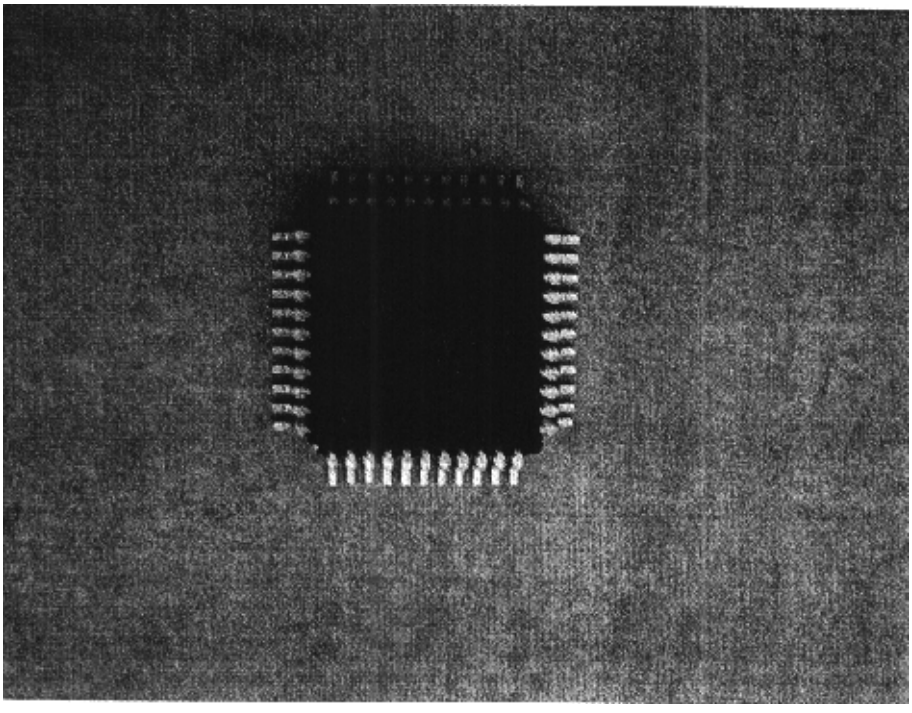
作为生产中直接使用的光刻版，除了包含与电路一致的电路图形外，还应包含多种辅助图形，这些辅助图形有分步取正图形、光刻对中图形、宽窄监测图形、电路陪测图形、划片刻痕图形，还可以表示有掩膜序号、焊块标记、芯片型号、厂家商标、版号日期等等的辅助图形。这些辅助图形应在版图验证完成之后加到版图中去。

5．7 本章小结

本章主要介绍了芯片版图的实现过程。生产线工艺我们采用了 0.6um 的硅栅工艺，设计规则采用首钢 NEC 的 CZ5L 工艺规则。

主要通过标准单元的设计方法，自动布局布线的设计流程完成了芯片版图的绘制工作。并详细介绍了输入、输出电路逻辑端口的结构，着重介绍了芯片高压输出驱动电路部分和输出掩膜电阻部分，给出了高压晶体管和高压电阻的设计规则，这在同类 VFD 专用驱动芯片中是具有创新意义的，避免了繁琐的外围电阻电路。

芯片实物图



VFD 专用驱动芯片实物图

该芯片现已通过浙江省电子产品检验所检验测定。

致　谢

衷心感谢我的导师孟利民教授，在我攻读研究生期间，给予我悉心的指导和关心。导师渊博的学识，高昂的工作热情，严谨求实的治学作风，积极吸收新鲜事物的精神给我留下了深刻的印象，将给我以后的学习和工作起模范和激励作用。近三年的学习期间，导师为作者提供了很好的学习和工作条件，认真耐心地解答了作者在工作学习中遇到的问题，值此论文完成之际，首先向导师孟利民表示衷心的感谢和深深的敬意。

衷心感谢周建国高工、徐洪富高工、楼向雄教授，在芯片设计期间，是你们给予了我实际工程项目中所需要的帮助和具体的指导，使我更好的把理论与实践结合在一起，加深了理解。

衷心感谢整个芯片设计组的老师、同学，感谢他们在方方面面给我的帮助和支持。

衷心感谢默默关心我的家人和朋友们。

参 考 文 献

[1] 甘学温，数字 CMOS VLSI 分析与设计基础，北京大学出版社，1999 年 2 月

[2] 夏宇闻，Verilog 数字系统设计教程，北京航空航天大学出版社，2003 年 7 月

[3] 张明，Verilog HDL 实用教程，电子科技大学出版社，1999 年 11 月

[4] 阎石，数字电子技术基础（第四版），高等教育出版社，1997 年 12 月

[5] 金西，VHDL 与复杂数字系统设计，西安电子科技大学出版社，2003 年 3 月

[6] 高保嘉，MOS VLSI 分析与设计，电子工业出版社，2002 年 12 月

[7] 杨之廉、申明，超大规模集成电路设计方法学导论，清华大学出版社，1999 年 3 月

[8] 洪先龙、严晓浪、乔长阁，超大规模集成电路布图理论与算法，科学出版社，1998 年 10 月

[9] 刘丽华、辛德禄、李本俊，专用集成电路设计方法，北京邮电大学出版社，2000 年 7 月

[10] Thomas & Moorby's Fourth Edition，The Verilog Hardware Description Language，刘明业、蒋敬旗、刁岚松等译，硬件描述语言 Verilog（第四版），清华大学出版社，2001 年 8 月

[11] 冯柏军、邸鹏、韩继国，MCU 内嵌 ROM 版图设计的程序实现，集成电路应用，2002 年 4 月

[12] 沈雷，CMOS 集成电路原理及应用，光明日报出版社，1993 年 3 月

[13] 居水荣，3V/5V 标准单元混合使用的芯片设计，半导体情报，2000 年第 2 期

[14] 李丽、高明伦、刘聪，芯片设计中读周期的同步问题，微电子学与计算机，2001 年第 4 期

[15] 徐勇、徐志军，一种 LCD 驱动控制芯片设计研究，电子产品世界，2002.1.B

[16] 居水荣、郑明，0.9um ASIC 正向设计中的总体仿真及测试矢量产生，微电子技术，2000 年 4 月，总第 132 期

[17] 蹇彤，9×64 位译码器专用集成电路的芯片设计，半导体技术，1997 年 12 月第 6 期

[18] 胡弈明、王枢华，LCD 动态显示驱动芯片 MAX7231 原理与应用，国外电子元器件，2001 年第 10 期

[19] 王珩，ASIC 的可测性设计，计算机与数字工程，1996 年第 5 期

[20] 邵志标、向凌顶、林长贵、朱秉升，IC 设计中晶体管直流模型参数的提取，西安交通大学学报，1997 年 1 月

[21] 张阳安、张阳天、王庆海、李玲，VFD 显示模块以及同单片机的接口，微计算机信息，2000 年第 16 卷第 6 期

[22] 邓海飞微电子学研究所设计室，Cadence 使用参考手册，2000 年 7 月

[23] IC/ASIC Design Composition and Analysis,Training Manual Version 4.3,Cadence Educational Services Group (ESG) March 18,1994

[24] HDL Synthesizer and Optimizer, Training Manual Release 4.2.1, Cadence Educational Services Group (ESG) February 18,1993

[25] Introduction to VHDL,Training Manual Version 1.2, Cadence Educational Services Group (ESG) April 19,1994

[26] Cadence Verilog Language And Simulation Training Manual Version 3.0,June 1999

[27] Cadence Verilog_XL Reference, Product Version 2.7,October 1998

[28] Cadence Library Manager User Guide,Product Version 4.4.6,June 2000

[29] Envisia Silicon Ensemble Place-and-Route Reference, Product Version 5.3,April 2000

[30] Virtuoso Schematic Composer User Guide, Product Version 4.4.6,June 2000

[31] Virtuoso Layout Editor User Guide, Product Version 4.4.6,June 2000

[32] 王鸿猷、许文俊，Introduction to Dracula Verification Training Manual Ver 1.0,June 2000

[33] Douglas J. Smith, HDL Chip Design, Doone Publication 1998

[34] IEEE Design Automation Standard Committee, IEEE, Standard hardware Description Language Based on Verilog® Hardware Description Language ,IEEE 1996

[35] Eli Sternheim et al, Digital Design and Synthesis with Verilog HDL, Automata Publishing Company, 1993

[36] Actel HDL Coding Style Guide,1997

[37] Farzad Nekoogar："Timing Verification of Application-Specific Integrated Circuits (ASICs)"，Prentice-Hall INC,1999

攻读硕士学位期间发表论文

本人已在国内核心期刊上发表论文三篇：

1. 杨骞、孟利民，综合宽带数据网的技术特点及其发展，中国有线电视，03 年第 4 期
2. 杨骞、孟利民，智能光网络技术及其发展，光通信技术，03 年第 3 期
3. 杨骞、孟利民，基于 HFC 网络的视频点播系统（VOD），电视技术，03 年第 8 期