版权声明

摘 要

随着集成电路特征尺寸的不断缩小，集成电路的功耗、面积、时序等都将面临着更加严峻的挑战。随着技术的进步，越来越多的芯片不断追求低功耗、高频率，而减小芯片的面积更是降低芯片制造成本的关键之一，在集成电路设计中，将遇到越来越多新的难题。如何运用新的技术去解决 40nm 甚至以下工艺的设计难题，是我们需要去学习和探讨的。随着电子产品更新换代的速度不断加快，对于芯片设计公司而言，如何提高产品上市速度，更好、更快的完成芯片的设计，也是我们需要去积累和摸索的。

本文结合利用 TSMC 40nm 工艺制造的一款芯片的后端设计为例，结合 Cadence 公司的 EDI、ETS、QRC， Synopsys 公司的 Prime Time 等 EDA 工具，详细介绍了基于 Encounter 的集成电路后端设计流程，对此流程中的每一个阶段所需完成的主要工作也做了简单的介绍。

本文详细介绍了集成电路后端设计的流程，重点介绍了静态时序分析方面的相关原理及解决时序违例的方法，并结合实际芯片设计过程中遇到的问题进行了分析和讨论。同时，对于如何完成布图规划、如何预防和消除串扰、在芯片流片之前所应完成的签核验证技术，也都做了相应的分析和介绍。同时，同于 40nm 工艺条件下，需要遵守的一些特殊的设计规则，也做了归纳和总结。

关键字： 后端设计、流程、静态时序分析、布图规划、信号完整性

Abstract

IC has developed with a high speed and high performance in latest several years. With the continuous shrink on feature size and the continuous improvement on integration and speed, the power of IC is becoming increasingly high. Die size, timing, power and so on, are becoming more and more critical and challenging. How to utilize new technology to solve these challenges with 40nm or even following process, that's what we need to research and explore. With the advances in technology, the electrical products have updated with a higher speed, but at the same time, the chip design company is facing a higher pressure on cost and time-to-market. How to design a chip with a higher performer and shorter period is also what we need to research.

In this paper, according to introduce one chip's back-end design, which is based on TSMC's 40nm manufacturing process and EDA tools, such as EDI, ETS, QRC from Cadence and PT from Synopsys, I have give a detailed description on EDI's foundation flow, and at the same time, I have also introduced what the EDI tool will do at each step under the foundation flow.

As for the static timing analysis (STA), I have described the relevant solutions on how to fix the setup and hold violations. Meanwhile, on how to tune the floor plan and how to prevent and fix noise, you can also find my analysis and introduction.

Key Word: Back-End Design, EDI's Foundation Flow, Static Timing Analysis, Floor Plan, Signal Integrity Analysis

目　录

图 目 录

表 目 录

第一章 绪论

1.1 集成电路后端技术的发展和现状

随着集成电路特征尺寸的不断缩小，集成电路的集成度和速度不断提高[1]。集成电路后端技术与工艺技术的关系相当紧密，在各大工艺厂商的技术努力和半导体器件材料研究人员的研究下，半导体工艺技术日新月异。最近 10 年，工艺节点迅速从 0.5um 降低到现在的 45nm、40nm。目前市面上的芯片主要涵盖了从 90nm 至 40nm 的工艺制程，而较为先进且对工艺依靠度较高的芯片（例如 CPU 芯片、图形显示芯片）已经在 40nm 实现了量产。

国外先进水平的公司目前已经可以大规模设计 40nm 和 28nm 的芯片，并已成功生产上市，AMD 也已经推出了 28nm 的 GPU。而国内大部分公司的水平尚且处于 90nm-65nm 阶段，并且在低功耗设计方面和国外先进公司仍然存在较大的差距，芯片规模一般不超过千万门级。

由于工艺非常先进，40nm 面临着很多挑战，如采用新型第二代硅锗应变硅(second generation strained silicon with Si-Ge)工艺，需要一个工艺库积累和工艺库验证的过程；湿浸式光刻技术也是业界第一次采用，在生产工艺成熟度方面需要经过验证。这些验证必须通过最后成品的缺陷率来反映，而不可能在生产前就非常清楚地预见到。如此精细的工艺，每一个工艺环节都会对最后的成品率产生直接的影响。此外，成品率的问题会使成本居高不下。虽然工艺每升级一次，就能使晶圆的制造成本降低一半，但是却会使芯片的开发成本上升 2 倍、3 倍甚至更多，而且是呈指数上升的。高工艺节点的研发费用更会高得惊人。其未来采用厂商只会越来越少，因为最先进工艺的 IC 设计只能由少数顶尖公司实现。

而对于工艺更为先进的 28nm 技术，台湾 TSMC 公司已于 2011 年 10 月份宣布，该公司已经开始为客户量产 28nm 工艺的晶元，相关的客户包括有 AMD，Altera，Nvidia，Qualcomm 以及 Xilinx。而这也让 TSMC 公司成为目前第一家推出 28nm 代工业务的公司。根据 TSMC 公司的介绍，目前流片 28nm 工艺产品的

客户是当前推出 40nm 工艺时的 2 倍。同时 TSMC 28nm 工艺在产能和产量上也全面超越了前一代工艺。另外就是 TSMC 公司的几个重要客户已经收到了基于 28nm 工艺的芯片产品。

28nm 和 40nm 工艺将会存在相当长的时间，根据计划 2012 年下半年 TSMC 将会推出 22nm 工艺，第一次推出将会在 2012 年 3 季度。在 22nm 及以后工艺，将会推出 2 种模式。第一就是会推出第二代 High-K Metal Gate。在初期会继续使用 193nm 沉浸光刻和双层图形，接下来将会向 EUV 或多重电子束(Multiple E-beam Direct Write)转换。

随着工艺的不断进步，芯片设计人员在后端处理过程中需要面对的问题也越来越复杂，从最初的仅仅需要“布局布线和时序验证等步骤”变成了现在的涵盖半导体各个方面的复杂工程[2]。

现在的后端设计通常需要面对下列问题：

1）设计的功能越来越复杂，时序越来越难以收敛；

2）芯片对低功耗的要求越来越严格，电源网络和低功耗设计成为重难点；

3）芯片规模变大，内部 IP 使用增多，布局布线复杂度呈几何增长；

4）工艺节点的降低带来了很多工艺生产上的问题，例如光刻精度降低、刻蚀几何尺寸变形等问题，提高了 DFM（Design For Manufacture）的重要性[3]；

5）给版图的验证和测试工作带来了新的挑战；

6）设计规模的提高带来设计周期的增长，而 TTM（Time To Market）直接决定了芯片厂商获取利润的周期[4]；

7）工艺节点的降低使得很多在 um 级工艺能够使用的器件模型不能继续使用，芯片的特征尺寸越来越接近材料的分子尺寸，需要重新建模并充分考虑 PVT（Process Voltage Temperature）的影响；

8）特征尺寸的降低使得各种连线的间距变得更近，互连线间串扰等信号完整性问题变得越来越突出[5]。

1.2 论文背景

本课题来源于某科研机构的 CPU 芯片，公司后端设计的主要任务是将获取的芯片的行为级代码，经过综合、静态时序分析、形式验证、布局布线、后仿、

DRC/LVS 等设计和验证步骤，将其实现为功能正确并且满足时序要求的芯片版图，提供给半导体生产厂商进行加工，最终得到可供实际应用的完整芯片。

此芯片的主要指标：

1）工艺：　　TSMC 40nm 1P9M

2）芯片工作频率：　2G/1.5G/500M

3）规模：约 200 Million Gate Count

4）面积：　　21600um X 21355um

在此项目中，本人负责其中的一个模块（BLOCK），主要参数如下：

1）面积：　　1850um X 2800um

2）时钟频率：　　500M

3）标准单元：　　约 30 万门

4）输入输出端口：　　1394

5）存储器（Memory）：　　162

6）寄存器（FlipFlop）：　48644

本人的主要工作任务如下：

1）基于 TSMC 40nm 1P9M 工艺库，基于 Cadence EDI 工具后端设计的流程，完成布图规划（floorplan），并根据静态时序分析的结果调整布图规划，以更好的优化时序；

2）布线完成之后，完成整个模块的静态时序分析，并修复所有的时序违例；

3）完成整个模块的信号完整性分析，修复串扰和噪声；

4）完成整个模块的设计规则检查（Design Rule Check）和 LVS(Layout Versus Schematic)验证，并修复所有的 DRC 违例；

5）完成整个模块的签核（Sign-Off）检查。

1.3 本文的组织结构

本文组织结构是按照整个后端设计的流程进行的。首先是利用 EDI 工具建立 database，然后调整布局规划（floorplan），物理综合（physical synthesis），进行时序分析和优化，信号完整性分析，设计规则检查（Design Rule Check），签核（sign-off check），　出片（Tape Out）。

本论文的总体结构如下：

第 1 章的主要内容是介绍了当前集成电路后端技术的发展和现状；介绍了本文的背景以及本人的主要工作；对论文的整体结构做一概述。

第 2 章介绍基于 Cadence Encounter 的数字集成电路后端设计的流程；对此流程中的每一个步骤，包括布图规划和布局布线、时钟树的设计和综合、静态时序分析、信号完整性分析等理论和关键技术进行简单的介绍。

第 3 章介绍在芯片后端设计布图规划的过程中需要考虑的问题及如何调整布图规划以满足静态时序分析方面的要求。

第 4 章详细介绍静态时序分析的理论；分析造成时序违例的因素以及相应的解决方法；并结合芯片设计中遇到的具体时序违例的路径进行分析。

第 5 章详细介绍串扰信号产生的原因；避免信号串扰所采取的预防措施以及对串扰的修复方法；介绍 40nm 工艺条件下需要遵守的一些设计规则以及对应的解决方法；介绍设计规则检查（Design Rule Check）以及 LVS(Layout Versus Schematic)验证的检查方法；介绍 EDA 相结合的签核（Sign Off）技术。

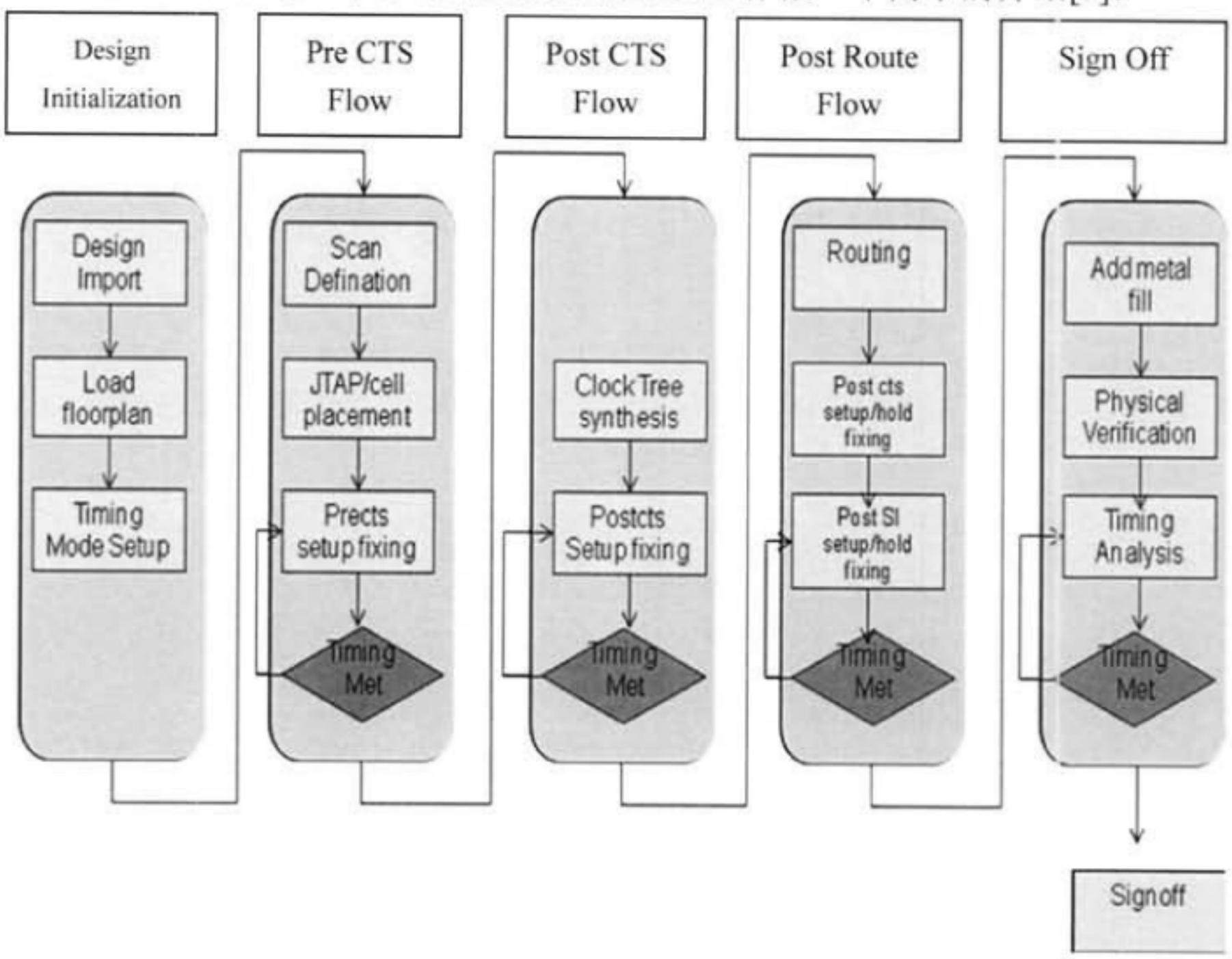
第 6 章是总结和展望，主要是对全文工作的总结，以及目前正在进行的工作，下一步需要研究的重点和将来工作的展望。

第二章 集成电路后端设计流程

2.1 基于 Encounter 的集成电路后端设计流程

传统的大规模集成电路设计流程中，逻辑综合之后的步骤都属于后端设计，而且前后端设计是相互独立的[6]。后端设计的流程，简单来说，就是将从客户手中拿到的 RTL 级网表，经过物理综合、时钟树综合、静态时序分析、布局布线、形式验证、后仿、DRC/LVS 等设计和验证步骤，最终将其实现为功能正确并且能满足时序要求和设计要求的芯片版图，提供给 TSMC 等半导体厂商进行流片，最终得到满足各方面既定指标、能够正常工作的实际芯片。

下面通过 Cadence 公司的一款后端设计工具 Encounter 的流程图，对后端设计的大体流程以及每一个步骤所要完成的主要工作做一个简单的介绍[7]：



图表 1 图 2-1 基于 EDI 的后端设计流程图

整个流程大致分为 5 个阶段[8]：

1)建立 database 的阶段：

2)Pre CTS 阶段；

3)Post CTS 阶段；

4)Post Route 阶段；

5)Sign-off 阶段。

在 database 的建立阶段，我们需要用到以下数据，包括：网表、布图规划的文件（floorplan）、库文件（包括所使用到的标准单元库、各种类型的存储器的库文件、各种 IP 相关的库文件等）、标准时序约束文件（SDC）、lef 文件、时钟树综合的说明文件等。

其中时钟树综合的说明文件，主要用于定义时钟树综合的相关参数信息，包括以下几个部分：时钟周期、max/min skew、max/min delay、最大扇出的数目、进行时钟树综合的时候所要用到的 clock buffer 的大小和类型等信息[9]。下面所示即是本人所负责的 BLOCK 所用于进行时钟树综合文件中对其中一个时钟的定义：

```
#-------------------------------------------------------
# Clock Root     : cru_io_clk
# Clock Name     : clk_io
# Clock Period : 2ns
# Clock Name     : clk_io
# Clock Period : 2ns
# Clock Name     : clk_io
# Clock Period : 2ns
#-------------------------------------------------------
AutoCTSRootPin cru_io_clk
Period              2ns
MaxDelay            0。01ns # sdc driven default
MinDelay            0ns # sdc driven default
MaxSkew             50ps # sdc driven default
SinkMaxTran         80ps # set_clock_transition
BufMaxTran          80ps # set_clock_transition
Buffer              CKBD4BWP12T CKND12BWP12T CKND16BWP12T
MaxFanout           12
NoGating            NO
DetailReport        YES
#SetDPinAsSync    NO
#SetIoPinAsSync NO
#SetASyncSRPinAsSync    NO
#SetTriStEnPinAsSync NO
#SetBBoxPinAsSync NO
```

```
RouteClkNet      NO
PostOpt          YES
OptAddBuffer     YES
RouteType        specialRoute
#LeafRouteType   regularRoute
END
```

从文件中可以看出，这个时钟的频率为 500M 赫兹；我们所定义的最大扇出的数目为 12；进行时钟树综合所使用到的 clock buffer 的类型包括以下几种：CKBD4BWP12T CKND12BWP12T CKND16BWP12T。其中对于最大扇出数目的设定，并无具体要求，根据经验，最大扇出设定为 12~18 之间。如果扇出过大，由于不同负载并不一定处于同一区域，会造成距离比较远的负载连接到时钟树的走线过长，线延迟增大，造成较大的时钟延迟，影响时序结果。

在 Pre CTS 阶段，完成的工作主要包括标准单元的放置、时钟树综合之前时序的优化[10]。EDI 工具会根据用户所定义的 floorplan、时序约束文件（SDC）、以及时钟树综合的说明文件，对整个 BLOCK 所包含的标准单元进行放置。对于大规模设计来说，整个芯片的布局规划在很大程度上决定了芯片时序性能和布线复杂度的高低。一个好的布局规划可以让相应的模块联系更紧密，缩短彼此连线，压缩关键路径长度，以及提高布线通道的利用效率。同时，布局规划过程中还需要完成对宏单元（Macro）的放置，主要包括各种 IP，比如 SATA、USB、PLL 等 IP，以及 BLOCK 内部所用到的 memory 单元。合理的宏单元摆放能够优化端口位置、缩短互连长度。此外，因为模拟电路对噪声和干扰的敏感性，对模拟宏单元的摆放也应该特别考虑。EDI 工具在进行标准单元放置（placement）这个动作的过程中，会充分考虑到时序的影响，比如，属于同一个模块（module）的标准单元会放置在一起，相互之间有时序关系的标准单元也会尽量放置在附近；这样在时钟树综合之前就充分考虑到放置标准单元对整个 BLOCK 时序的影响。

由于寄存器等时序器件的普遍使用，时钟信号在数字电路中扮演着举足轻重的作用。但是也正因为时钟信号遍布芯片范围的各个角落，使得各个位置寄存器的时钟信号的相位间存在了较大的差异。这种差异对于同步电路来说是致命的，因为它会导致各种情况的时序违规的发生，因此我们需要对时钟树进行综合，以使其满足全局延迟均衡的要求。所以在 Post CTS 阶段，主要完成时钟树的综合（根据时钟树综合的说明文件中的参数要求完成）以及时钟树综合完成之后时序

的优化（在这个阶段，EDI 工具主要对 setup 违例的时序路径进行优化，包括输入端口到寄存器（in2reg）、寄存器和寄存器之间（reg2reg）、寄存器到输出端口（reg2out）以及输入端口到输出端口（in2out）这四种类型的时序路径，尽可能地减少布线拥塞和时序收敛的迭代次数[11]。在这一阶段，EDI 工具并不会对 hold 违例的时序路径进行优化或者修复。

在 Post Route 阶段，主要完成布线、进一步的 setup/hold 时序优化。在优化时序的过程中，对噪声同样进行一定程度的优化。在这个阶段，用户可以根据 EDI 工具生成的时序报告，来评估前一阶段的布局规划是否合理、时钟树综合是否满足要求，从而决定是否需要进一步调整布局规划和重新对时钟树进行综合、调整。在布局规划还没有完全确定下来之前，我们需要不断的分析时序报告，并根据分析结果对布局规划做出相应的调整，然后重新进行前面提到的 Pre-CTS、Post-CTS、Post Route 三个阶段，直到用户认为时序已经在可控的范围之内，在继续进行一些手动的修复工作。

在 Sign-off 阶段，主要完成设计规则检查（Design Rule Check）、物理验证（Physical Verification）等工作，并进一步检查在完成所有的设计规则检查和物理验证的工作之后，不会出现新的 setup/hold 时序违例。

在整个流程中，标准单元的放置、时钟树的综合以及布线都是由工具根据用户所定义的相关参数和选项，自动完成；布图规划需要人为设计，并根据时序分析的结果不断进行调整；EDI 工具会优化一部分的 setup/hold 时序为例，其他的时序违例则需要工程师结合工具进行修复。因此论文的重点将集中在介绍如何进行合理的布图规划、如何解决时序违例问题、如何修复串扰、如何进行设计规则的检查、物理验证以及签核（sign-off）技术。由于时钟树对整个芯片时序收敛的重要性，论文中也会对时钟树的综合的相关理论、方法进行详细的介绍和研究。

2.2 后端设计中使用的 EDA 工具

EDA 技术的出现大大地提高了单位时间内设计者的设计工作量，刷新了设计人员的工作效率，也正是由于 EDA 技术的出现，才使得超大规模集成电路设计变得容易。从前端到后端，从测试到验证，从仿真到版图设计，几乎每个设计阶段都有很多的 EDA 工具供设计者挑选使用。下面简要介绍后端设计的流程中

所用到的主要后端 EDA 工具。

一、Synopsys 公司的 Prime Time

Prime Time 是 Synopsys 公司的用于进行静态时序分析软件。Prime Time 适用于门级电路设计，可以和 Synopsys 公司的其它 EDA 软件非常好的结合在一起使用，通常也是作为后端设计签核（sign-off）阶段的时序验证的标准工具[12]。

作为专门的静态时序分析工具，Prime Time 可以为一个设计提供以下的时序分析和设计检查：建立和保持时间的检查(setup and hold checks) ；时钟脉冲宽度的检查；时钟门的检查(clock-gating checks)；非时钟驱动的寄存器检查（unclocked registers）；未约束的时序端点检查(unconstrained timing endpoints)；组合反馈回路检查(combinational feedback loops)[13][14]。

使用 PT 对一个电路设计进行静态分析，一般要经过以下几个步骤：设置设计环境；指定时序约束；设置时序例外；进行时序分析[15]。

Prime Time 作为一款经典的静态时序工具[16]，随着集成电路规模的不断增大，其处理大规模集成电路的能力也不断提高，能很好的适应大规模集成电路 SOC(system-on-chip)的设计。

二、Cadence 公司的 Encounter

SOC Encounter 是 Cadence 公司针对大型芯片快速设计出片所设计的一款软件[17]。在芯片大小快速成长的今天，一个建立在 65nm、40nm、28nm 工艺制程之上的千万门逻辑单元的设计已经非常常见。面对这样的复杂度以及日益增加的 time-to-market 的压力之下，设计者需要一个快速且可靠的系统。

SOC Encounter 使用一连串的收敛方式帮助使用者缩短设计时间以及减少反复次数，这一连串的收敛方式起始于硅晶片的实质规划，一种初始的整体芯片连线表示方式[18]。这个实质规划考虑到设计的所有角度，诸如逻辑、时序、信号完整性、电压下降、电子漂移以及输入输出同时动作产生的影响。群组设计可借此分析结果之效能，并由各个独立的设计者完成各部分，然后将之整合起来，重新分析整个设计。

SOC Encounter 完整提供了由合成、布局到布线的系统解决方法，其中包括了快速实现规划、阶梯式设计切割划分、芯片平面布置、实体合成、详细绕线连接以及信号完整性分析等功能，引导设计者快速且正确地实现其设计。09 年下

半年，Cadence 公司已经推出了出支持 28 纳米工艺节点的设计解决方案。

前面已经详细介绍了基于 Encounter 的后端设计流程，这里就不再对这款软件的特点和功能做详细的介绍。

三、Cadence 公司的 QRC 和 ETS

Cadence QRC Extraction 可为基于单元的数字设计提供超越其他撷取技术的、有制造意识的硅晶密度[19]。透过其分布于多重网络处理器和运算集群的近似线性性能伸缩，它可显著减少处理时间。它还为 Cadence Encounter 数字 IC 设计平台提供强大的多边际条件支持和精确的增量式基于设计提交（sign-off）的参数撷取。

Cadence 的产品分割策略针对特定等级的设计复杂性，为客户提供多种等级技术。Cadence QRC Extraction 有 L、XL 和 GXL 系列产品可供选择。

在此项目之中，我们主要使用了以下工具：

RTL 代码综合工具：	Design Compiler	（Synopsys）
数字仿真工具：	ModelSim	（Mentor）
静态时序分析工具：	PrimeTime	（Synopsys）
功耗分析工具：	RedHawk	（Apache）
形式验证工具：	Formality	（Synopsys）
布局布线工具：	Encounter	（Synopsys）
寄生参数提取工具：	QRC	（Synopsys）
DRC/LVS 工具：	Calibre	（Mentor）
版图修改工具：	Virtuoso	（Cadence）

第三章 后端设计的布图规划

3.1 布图规划的目标

布图规划对于芯片设计的重要性是不言而喻的。布图规划的合理与否，直接关系到芯片的时序收敛、布线通畅、电源稳定以及良品率[20]。一个合理的布图规划可以大量减少时序违例的产生，充分提高时序收敛速度。规模超过上百万门的集成电路后端设计，往往需要几十次的布局布线迭代，才能找出最优方案[21]。

布局又称为标准单元放置，它实际上包括对 I/O 单元的排序放置、大模块（block）放置和标准单元放置的规划。在布图规划开始时，首先要准备好各种基本设计数据和相应的物理库、时序库文件，并输入到布图规划的工具环境中来，为其后的布局和布线做好准备。

布图规划的主要内容包含了对芯片大小的（die size）的规划、芯片设计输入输出（I/O）单元的规划、大量硬核或模块（hard core、block）的规划等。在一些较为复杂的超大规模集成电路设计中，为了尽量减小时钟信号线的偏差，在布局之前就需要对时钟网络进行规划[22]。

布图规划的目标主要可以分为以下 3 个部分：

1)确定芯片的面积。

出于成本的考虑，芯片面积越小，每张硅片（wafer）上产出的裸片（die）数量将增大，从而平均到每个芯片上的成本会降低，但是如果布图规划设定的裸片面积太小，则会造成拥塞程度高，难以布线。

在后端设计中，通常会根据整个芯片的尺寸和布图规划来估计每个模块(block)的尺寸、形状和位置。各个模块再根据自己的布图规划的静态时序分析结果和利用率（Density）来评估模块的尺寸和形状是否需要做出调整. 输入输出端口的位置是否需要调整。最终根据整个芯片和各个模块的评估结果来确定各个模块最终的尺寸、形状和输入输出端口的位置。

2）确保时序的收敛。

在集成电路设计中，芯片的所有功能都是在时钟控制下完成，所以从一个寄存器到达另一个寄存器的路径长短决定了芯片的性能。在芯片设计的布图规划阶

段就要充分考虑到最终芯片是否满足设计的标准时序约束（SDC）的要求，实现时序的收敛。因此在布图规划阶段就需要对芯片的延迟进行预估。在此项目中，我们是使用 Cadence 公司的 EDI 工具进行布图规划的。在布图规划的各个阶段，EDI 工具都会产生相应的时序报告。表 3-1、3-2、3-3 所列出的分别是 preCTS、postCTS、postRoute 三个阶段的时序报告，通常我们就是根据这些报告来评估我们每一步的设计是否合理：

表格 1 表 3-1 prects timing report

Setup mode	all	reg2reg	in2reg	reg2out	in2out	clkgate
WNS (ns):	-1.035	-0.005	-0.938	-0.042	-1.035	1.711
TNS (ns):	-28303.6	-0.022	-28275.1	-0.187	-28.342	0.000
Violating Paths:	43801	8	43749	16	28	0
All Paths:	1.2e+05	1.19e+05	46449	870	28	3

DRVs	Real		Total
	Nr nets(terms)	Worst Vio	Nr nets(terms)
max_cap	1 (1)	-0.004	44 (44)
max_tran	77 (198)	-0.276	81 (202)
max_fanout	5 (5)	-7	48 (48)

Density: 52.195%
Routing Overflow: 0.02% H and 0.17% V

表格 2 表 3-2 postcts timing report

Setup mode	all	reg2reg	in2reg	reg2out	in2out	clkgate	default
WNS (ns):	-1.171	-0.012	-1.171	-0.041	-1.019	1.269	0.000
TNS (ns):	-42389.8	-0.028	-42361.1	-1.030	-27.624	0.000	0.000
Violating Paths:	44192	6	44099	59	28	0	0
All Paths:	1.2e+05	1.19e+05	46505	870	28	3	0

DRVs	Real		Total
	Nr nets(terms)	Worst Vio	Nr nets(terms)
max_cap	0 (0)	0.000	0 (0)
max_tran	114 (287)	-0.201	119 (309)
max_fanout	30 (30)	-30	1717 (1717)

Density: 53.539%
Routing Overflow: 0.06% H and 0.70% V

表格 3 表 3-3 postroute timing report

Setup mode	all	reg2reg	in2reg	reg2out	in2out	clkgate	default
WNS (ns):	-0.869	-0.012	-0.869	-0.136	-0.862	1.257	0.000
TNS (ns):	-31303.1	-0.154	-31273.0	-6.429	-23.455	0.000	0.000
Violating Paths:	44209	31	44019	131	28	0	0
All Paths:	1.2e+05	1.19e+05	46505	870	28	3	0

DRVs	Real		Total
	Nr nets(terms)	Worst Vio	Nr nets(terms)
max_cap	0 (0)	0.000	0 (0)
max_tran	28 (88)	-0.012	39 (152)
max_fanout	4 (4)	-35	1691 (1691)

Density: 54.915%

从上面的三份时序分析报告可以看出，在这三个不同的阶段，输入端口到寄存器存在较大的时序违例，并且总的时序违例的数目还相当多，说明在布图规划的时候，有可能存在寄存器放置的位置离输入端口过远，造成较大的互连线延迟，从而产生较大的时序违例；另外一种可能就是客户设定的 SDC 中，对某些输入端口的输入延迟值（input delay）设定过高，很难满足时序的要求。对于前一种情况，需要把相关的寄存器重新放置到输入端口附近，或者通过利用 EDI 工具设定一个区域（guide or region），把通过这个输入端口的所有负载，限定在这个区域内。不过在考虑 in2reg 时序结果的同时，我们必须同时考虑到与这些寄存器相关的 reg2reg 时序是否会受到影响。

3）满足布线的要求。

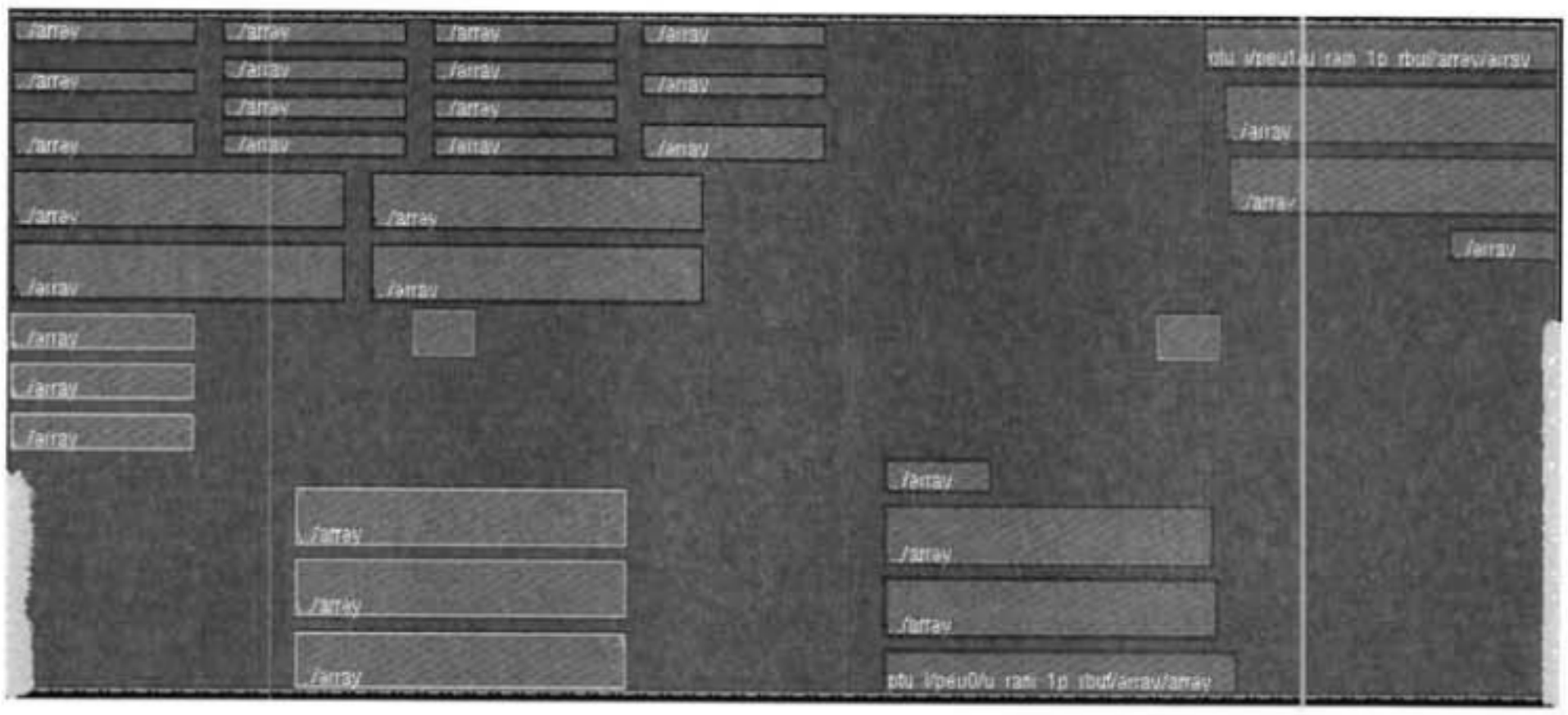
布图规划与布局完成了芯片的器件（包括芯片中所使用到的各种存储器、IP 等）摆放，而芯片功能的实现则需要将所有的器件按照一定的要求连接在一起，而布图规划的目的是为了方便走线，在保证布线通畅的同时，尽量缩短走线的长度，也即减小了互连线的延迟[23]。在布线的过程中，我们应尽量避免布线拥塞的发生，同时还要尽可能的减小互连线长度，以免产生较大的互连线延迟，对时序性能造成较大的影响。

在实际布图规划的过程中，主要基于以上 3 个方面对模块的布图规划做出调整，并尽可能满足以上 3 个方面的要求。在本章，将详细介绍在本模块实际布图

规划中需要考虑的问题（包括基于设计规则检查（DRC）、ESD 保护、串扰避免等因素的考虑），遇到的难题（包括时序方面、布线拥塞方面等）以及相应的解决方法（包括提前放置相关的标准单元，指定相关子模块中标准单元的位置和区域等）。

3.2 布图规划方案的确定

图 3-1 即为此项目中本人所负责的一个 block 的布图规划的方案：



图表 2 图 3-1 布图规划

1）Memory 位置的确定。

在图 3-1 的 floor plan 中，所有的 memory 使用了 4 种颜色进行了标记，分别代表此 block 中所包含的 4 个 module。从图 3-1 可以看出，我们把属于同一个 module 的 memory 集中放在了一起。

右上角红色边框的 memory，属于 module peu1；右下方暗红色边框的 memory，属于 module peu2；因为这两个 module peu0 和 peu1 之间相互之间存在时序检查关系，所以我们在摆放与这两个 module 相关的 memory 的位置的时候，尽可能的把他们放在位置距离比较相近的地方，以避免距离过远，互连线延迟和 transition 时间会很大，有可能造成较多时序违例的产生。

左上角紫色边框的 memory 以及左下角黄色边框的 memory 同属于 module ptuul 之下的两个子 module，所以我们在摆放 memory 的时候，依然按照同一个 module 相关的单元摆放在一起的原则，就近进行摆放。

2）I/O 位置的摆放。

在 block 边界的左下方和右下方，黄色区域是摆放输入输出端口的位置。

I/0 位置的确定，主要是由整个芯片的布局规划决定的。一个完整的芯片布局，是需要考虑到各个 block 之间输入输出端口的对接，保证他们之间的互连线长度可以做到最短，尽可能的减少线延迟，保证 block 之间的时序路径延迟最短；同时要考虑到不会因为输入输出端口的对接不准确造成各个 block 之间有限的空间区域内布线的拥堵。对于 block 内部而言，在进行布图规划的时候，也要充分考虑到 in2reg 和 reg2out 这两组路径的时序要求，与输入输出端口存在关联的寄存器和组合逻辑单元，不能放置在离输入输出端口距离较远的地方。

3）是否需要提前摆放一些标准单元。

通常对于大规模的集成电路后端设计，都会对整个芯片进行一定的划分[24]，根据功能的不同，划分为不同的 block。但为了保证各个 block 之间的时序同样能够得到满足，我们会在时序约束文件中定义好各个输入输出端口的延迟值（input/output delay）。在进行布图规划的过程中，不仅要满足 block 内部的时序要求，也需要满足 block 之间的时序要求，这就要求我们考虑到输入输出端口与内部 flip-flop 之间的时序检查的结果，在 EDI 中，我们可以在每一步结束之后的时序报告中看到相关的结果，即对应的 in2reg(输入端口到 flip-flop 的时序路径)、reg2out(flip-flop 到输出端口的时序路径)以及 in2out(输入输出端口的时序路径)。通过分析结果来确定我们是把相应路径上的 flip-flop 及相应的组合逻辑单元放置在输入输出端口的附近还是通过插入一串驱动能力比较强的 buffer 或者 inverter 对，将输入输出端口连到相应的 flip-flop 上。

3.3 本章小结

本章主要介绍了布局规划阶段需要考虑的各种因素以及所要达到的目标。布局规划可以说是整个后端设计的最初阶段。一个布图规划的好坏，在相当大的程度上决定了时序能否收敛、布线是否会造成拥堵等一系列问题。所以在布图规划阶段，我们就要充分考虑到不同 module 的摆放位置对整个 block 时序的影响、对与其有时序关系的其他 block 之间的时序关系能否收敛。布图规划是一个需要不断调整、不断反复的过程。我们会在调整完每一版 floorplan 之后，继续进行后续的标准单元放置、时钟树综合、时序优化等阶段。采用 EDI 工具，在每一

阶段都会有对应的时序报告生成。我们需要分析每一阶段的时序报告，结合database，来分析布图规划存在哪些问题，是时序的问题、布线的问题还是寄存器与输入输出端口之间的时序存在问题。如果是时序的问题，我们要考虑是因为相关的 flip-flop 距离相隔过远，线延迟太大，造成没法满足时序。如果是布线的问题，我们要分析是哪一区域的布线存在问题，是因为这一区域摆放的标准单元太多，造成绕线的拥堵，还是因为 memory 之间的间隙太小，没有足够的空间走线。如果是输入输出端口与寄存器之间存在问题，我们就需要考虑我们是否需要把相关的 module 摆放到输入输出端口附近的位置，还是只需要把相关的一些 flip-flop 摆放在输入输出端口的附近。

总的来说，布图规划方案的确定对整个 block 时序的影响是至关重要的，在确定布图规划之前，我们一定要根据 EDI 工具所生成的时序报告，仔细分析原因，不断调整布图规划，直到整个 block 的时序不会出现大量的时序为例。

第四章后端设计的时序问题及解决方案

4.1 静态时序分析的重要性及其原理

4.1.1 静态时序分析的重要性

在集成电路芯片的物理实施中，完成布线之后的一项最重要、最必需的工作是进行静态时序分析[25]。静态时序分析贯穿于设计过程的各个阶段：从 RTL 逻辑综合，到布局、时钟树综合、布线和反标（back annotation），直到流片（tape-out）。每一次分析的目的都是为了检查当前设计的结果是否满足设计的约束条件。而最终的芯片能否正常工作，也是基于在静态时序分析阶段，我们是否发现并分析和修复了所有的时序违例。

我们之所以需要对芯片的设计流程中各个环节进行时序分析，一个最重要的因为是：半导体 MOS 管在理想状态下，当栅极电压达到开启要求的瞬间，电流可以立即从栅极下的沟道通过。但事实上，并不存在理想器件，所有的电流通过 MOS 管都有一个较短的时间（这个时间根据 MOS 管的工艺和结构的差异而有所区别；并且根据 MOS 管特性的不同，上升沿和下降沿的传输时间也是不同的）。当需要传输信号的时候，这个信号每经过一个器件，都会产生一定程度的延时，当这个延时达到一定程度的时候，就会对传输信号的稳定性和正确性造成较大的影响，从而使整个芯片的功能失效。

目前几乎所有的芯片中，都会用到时序逻辑器件，例如最常用的 D 触发器（D Flip-Flop）。这类时序逻辑器件对信号的稳定性和实时性有着极为苛刻的要求，过快或者过慢的信号传输都将造成正确数据的丢失。因此为了确保芯片的正常工作，在芯片设计的过程中，时序分析是非常重要的一环。

不同阶段进行分析的根本差别在于时序结果与设计约束之间相差的准确性。比如，在逻辑设计阶段做时序分析时，互连线参数是由 WLM 来粗略表示的，与实际结果会相差很大。在布局后做时序分析时，受到 I/O 和逻辑模块布图的约束，互连线参数可以比用 WLM 稍微准确的近似值表示。在时钟树综合后做时序分析时，设计的时序关键部分即与时钟相关的互连线参数已经很接近最终结果了。布

线后互连线 RC 最终固定成为实际值。

我们通常会在时钟树综合之后，根据时序分析的结果对布图规划或者时钟树的综合做出一定的调整，或者根据时序违例的实际情况，提前放置相关的标准单元或者模块。在时钟树综合之后的时序结果还算理想的情况下，继续完成布线，并需要分析 EDI 工具完成布线之后的产生的时序报告，进一步分析引起时序违例的原因，并找到相应的解决方法。

只有在完成布线之后，提取的互连线参数，在经过延时计算后才最接近设计的实际结果。这时来进行静态时序分析才有最可靠准确的，也具有最大的应用意义。

在静态时序分析中，我们假定设计中的所有时钟都是同步的。在分析时，根据逻辑关系，所有的路径都会受到检查。由于 EDA 技术的迅速发展，现在几乎所有的大规模集成电路芯片设计都会采取多模式多端角（MMMC， multi-mode multi-corner）进行更加快速的时序分析。而对于 TSMC 来说，他也会发布相应的说明文件，以告知客户，最终的静态时序分析结果需要包括哪些模式，哪些端角。以此项目的设计为例，最终 tape-out 之前，我们需要检查一共 8 个 mode、20 个 corner 下的时序结果。7 个 mode 包括：function mode；dft mbist mode；dft scan shift mode；dft scan lowspeed capture mode；dft scan highspeed capture mode；flush mode；intflush mode；jtag mode。20 个 corner 包括：

wc_cworst_125c_max ； wc_cworst_125c_min ； wc_rcworst_125c_max ；
wc_rcworst_125c_min ； wcl_cworst_m40c_max ； wcl_cworst_m40c_min ；
wcl_rcworst_m40c_max ； wcl_rcworst_m40c_min ； bc_cbest_0c_min ；
bc_rcbest_0c_min；bc_cworst_0c_min；bc_rcworst_0c_min；ml_cbest_125c_min；
ml_rcbest_125c_min ； ml_cworst_125c_min ； ml_rcworst_125c_min ；
lt_cbest_m40c_min ； lt_rcbest_m40c_min ； lt_cworst_m40c_min ；
lt_rcworst_m40c_min。

4.1.2 静态时序分析的原理

进行时序分析时，简单来说，就是将某一段路径的时序与时序约束的要求进行比较。时序约束文件有多种形式，其中标准设计约束（SDC）文件较为流行，

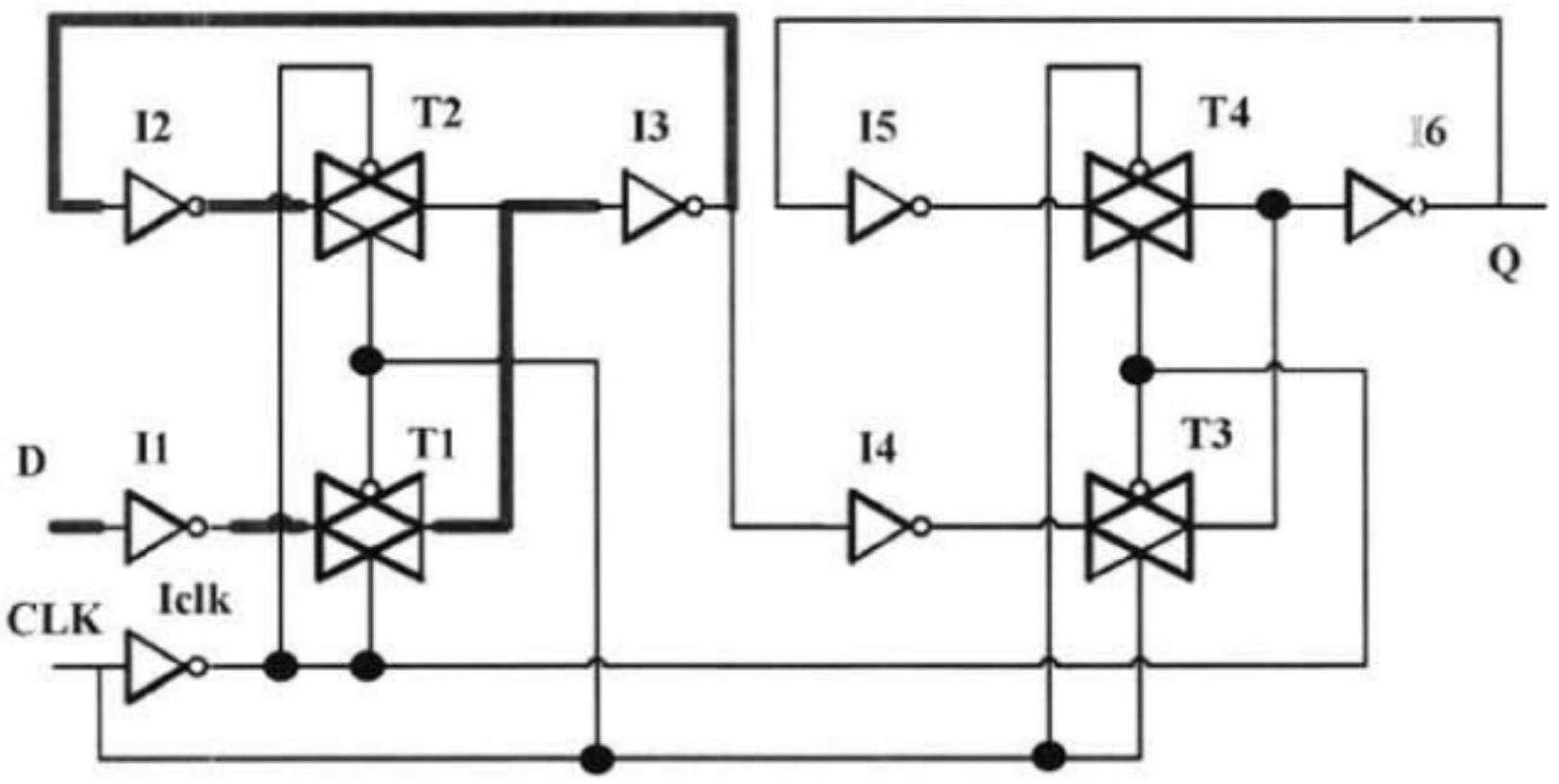
它是对整个设计时序要求的体现，贯穿于系统设计、逻辑设计以及物理设计的整个芯片设计流程。

对于触发器来说，每一段待分析的时序路径，都是由它的起点（start point）和终点(end point)来表示。时序分析的根本目的是为了检查在时钟的控制约束下，与其相关的数据能否符合时序要求被记录存储下来，这种时序检查就是常说的建立时间（setup）时序和保持时间（hold）时序。

Setup 的定义为：在时钟作用前沿（或后沿）到达前，同步输入信号（D）必须保持稳定的那段时间以使信号不至于丢失。

Hold 的定义为：在时钟作用前沿（或后沿）到达后，同步输入信号（D）必须保持稳定的那段时间以使信号不至于丢失。

接下来我们分析一下建立时间和保持时间的原理：

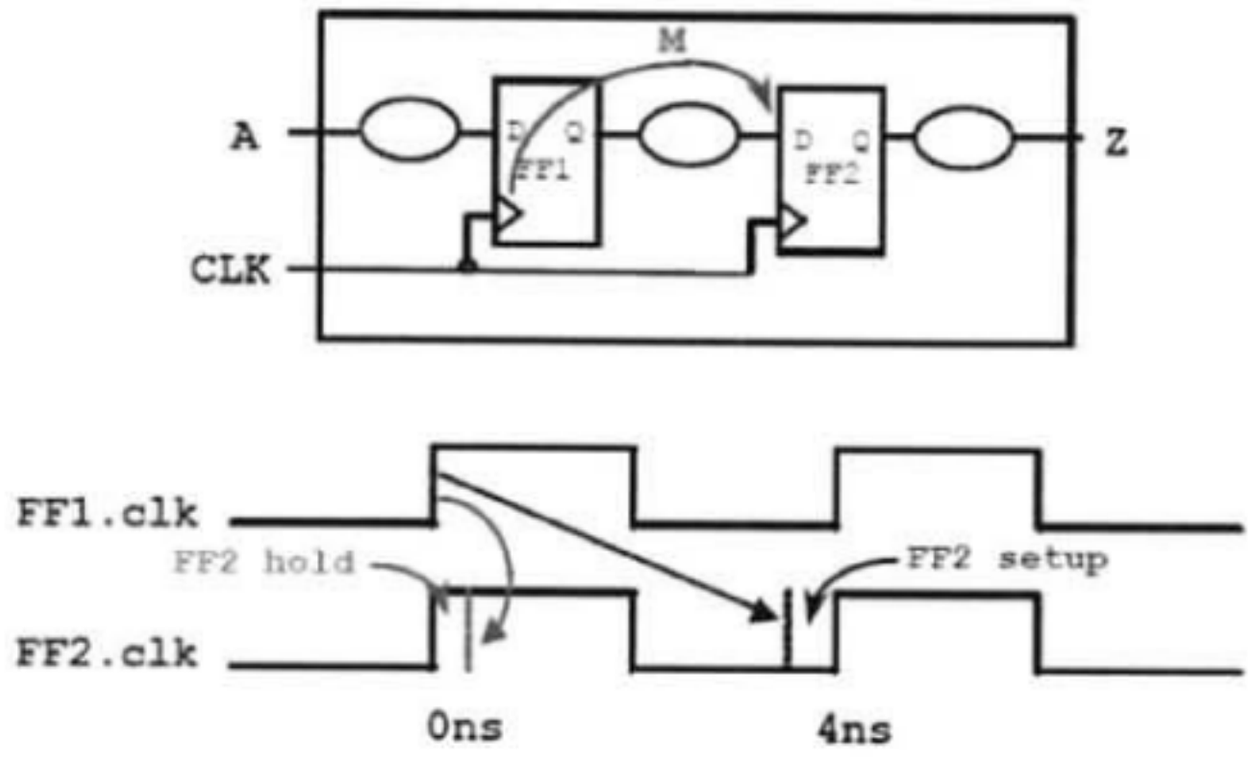


图表 3 图 4-1 D 触发器晶体管级原理图

当时钟信号 CLK 处于低电平的时候，传输门 T1 导通，T2 截止。为了在时钟信号 CLK 的上升沿到来之时能够顺利写入数据，则需要在时钟信号 CLK 的上升沿到来之后，端口 D 输入的数据已经在传输门 T2 的数据输入端做好准备。当时钟信号 CLK 上升沿到来的时候，T2 导通，数据信号即可传递至下一级。

图4-2中的红色加粗部分表示了触发器D端的输入信号在时钟上升沿到来之前需要传递的路径，而数据传输经过这一段路径所耗费的时间，就是所谓的器件的建立时间（setup time）。假设触发器 FF1 和 FF2 的器件延时分别是 T_{I1}、T_{I2} 等，则有如下要求：

$$T_{setup} = T_{I1}+T_{T1}+T_{I3}+T_{I2}-T_{Iclk}$$



图表 4 图 4-2 建立时间和保持时间的检测基准

图 4-2 说明，当 触发器 FF1 的 clk 上升沿到来的时候，数据从触发器 FF1 的数据输出端 Q 开始传输，经过一定的组合逻辑之后，到达触发器 FF2 的数据输入端 D，整条路径（Path）设为 M。如果要使得触发器 FF1 到 FF2 之间的数据传输时间能够满足建立时间（setup）的要求，则数据传输的时间 T(m)需要满足以下要求：

$$T_m + T_{FF2\_setup} < T_{period} = 4ns$$

相应的，如果要使得触发器 FF1 到 FF2 之间的数据传输满足保持时间（hold）的要求，则需要数据传输时间 T(m)满足以下要求：

$$T_m > T_{FF2\_hold}$$

但实际上，上面所描述的只是理想状态下的情况。在实际应用的过程中，时钟 1（FF1_clk）和时钟 2（FF2_clk）之间是不可能完全保持同步的。在器件的内部，也不可能在时钟上升沿到来的瞬间，就从数据输出端口 Q 发送出数据。考虑到这些非理想状况，需要引入以下参数来进行更准确的表述:

T_{clk1}：时钟上升沿从时钟源 CLK 传输到触发器 FF1 的时钟端口 clk 所需要的时间；

T_{clk2}：时钟上升沿从时钟源 CLK 传输到触发器 FF2 的时钟端口 clk 所需要的时间；

T_{skew}：$T_{clk2} - T_{clk1}$，代表两个触发器之间的时钟偏差；

$T_{clk\_to\_Q}$：触发器内部，时钟 clk 上升沿到来之时，到输出端口 Q 输出数据为止，所需要的时间；

T_{clogic}：信号传输经过两个触发器之间的组合逻辑（Combination Logic）所需要的时间。

则上式中的 T_m 可以表示成：

$$T_m = T_{clk\_to\_Q} + T_{clogic}$$

另外，考虑到每块芯片的工作环境是不尽相同的，甚至同一块芯片在不同时间段所处的工作环境也是不同的，这种环境的区别包括电压（voltage）、温度(temperature)、甚至半导体生产工艺 process 的区别（这也是通常所说的 PVT 条件）。半导体的基本物质 Si、Ge 的导电性能的好坏，均是与 PVT 有着密切关系的。这些因素，造成了芯片内部器件延时具有较大的不确定性。通常情况下，可以将器件的工作环境分为以下三种情况：

表格 4 表 4-1 芯片工作情况及其工作条件

Operation Condition	Voltage & Temperature
Best(Fast corner，器件速度快)	高压、低温
Normal(也叫 Typical)	常压、常温
Worst（Slow corner，器件速度慢）	低压、高温

考虑到以上这些因素，为了确保芯片在各种情况下都能够正常工作，给上面引入的三个新参数加注下标 min 或者 max，以此来体现建立时间和保持时间检测时的一些区别和需要注意的事项。在进行时序分析的时候，设计者们应该用最容易发生时序违例的情况（考虑最 worst 的时序情况）来进行对应的检测。例如，用 Best 的情况来检测保持时间违例，用 Worst 的情况来检测建立时间违例。

基于以上分析，建立时间需要满足以下要求：

$$T_{clk1} + T_{clk\_to\_Q\_max} + T_{clogic\_max} + T_{FF2\_setup} < T_{period} + T_{clk2}$$

将 $T_{skew} = T_{clk2} - T_{clk1}$ 代入上式可得：

$$T_{clk\_2\_Q\_max} + T_{clogic\_max} + T_{FF2\_setup} < T_{period} + T_{skew}$$

保持时间需要满足等式如下：

$$T_{clk1} + T_{clk\_2\_Q\_min} + T_{clogic\_min} > T_{clk2} + T_{FF2\_hold}$$

将 $T_{skew} = T_{clk2} - T_{clk1}$ 代入上式可得：

$$T_{clk\_2\_Q\_min} + T_{clogic\_min} > T_{skew} + T_{FF2\_hold}$$

在实际的芯片设计中，还需要考虑到时钟的不一致性。时钟网络对整个芯片

的重要性不言而喻。但同时，时钟信号也容易受到电磁干扰、环境变化等多种因素的影响，进而会对整个芯片的正常工作造成一定的影响。因此，在设计之初，给时钟信号一个准确、完备的描述，是很重要的事情，在进行时序分析的时候，给时序路径留下足够的余量（margin），对芯片的可靠性也是非常重要的。

4.2 造成时序违例的因素和解决方案

4.2.1 造成时序违例的因素

在集成电路芯片物理实施设计的过程中，几乎不可能不产生时序违例，尤其是在工艺节点不断降低，器件规模不多增大的情况下，时序违例的产生更是不可避免的。

造成时序违例的因素可以是一种或多种：

1）随着器件规模的不断增大，系统设计的复杂性不断提高，对于整个设计的约束设定来说，难度也不断增大，由此则不可避免的会遇到有些约束可能是不合理的，或许根本不可能实现[26]。这种情况是人为的约束设定不合理造成的。

2）有一些约束在逻辑综合时可能依据了不合理的 WLM，所产生的网表在物理实施时不可能实现。这种情况跟前端的设计的准确性有很大的关系。

3）由于互连线的相互影响，相互干扰，引起的时序违例。

4）由于设计者不合理的布局布线，使得物理综合后的设计，很难满足时序的要求。

在实际的设计中，最常遇到的时序违例情况，是上述的第三和第四种情况。所以，也要求后端工程师在设计之初，就要充分考虑到整个芯片的布局布线。

4.2.2 时序违例的解决方案

我们之所以要进行时序分析，在于在芯片制造之前就能够检查并发现问题，修复所有的时序违例，是芯片能够稳定正确的工作。时序检查中发现的违例主要有建立时间（setup）违例和保持时间（hold）违例两种。在大多数情况下，setup 的违例占主要成分，但随着芯片的规模逐渐增大，hold 违例的数目也不在少数。通常，我们会先解决 setup 的时序违例，再去解决 hold 的时序违例。但两者是相

互影响的。我们不可避免的会在解决 setup 时序违例的过程当中，引起新的 hold 时序违例；同样，在解决 hold 时序违例的过程当中，也有可能会引起新的 setup 时序违例。因此，这就要求我们在修复 setup 或者 hold 时序违例的时候，需要检查一下我们从这个点入手去进行时序违例修复的时候，这条时序路径是否有足够的余量（margin），从而保证我们在修复 setup 和 hold 违例的时候，不会引起或者尽可能少的恶化相应的 hold 和 setup 时序结果。

前面已经提到过，在 EDI 工具中进行物理综合、时钟树综合、时序优化的过程中，我们在每一个阶段都会进行相应的优化。也就是说，我们从后端设计的开始阶段，就需要充分考虑到时序方面对整个设计的影响。

比如，在放置标准单元（placement）这一阶段，我们会通过在 EDI 工具中设置下列选项来进行时序的优化:

setPlaceMode –timingDriven true

setOptMode –effort high

也就是说，在 EDI 工具进行自动放置标准单元的时候，它需要考虑到时序的影响，不能把两个相互之间有时序检查的模块或者触发器，放置到距离很远，或者布线不方便的两个地方。

在布线阶段，我们会设置：

setNanoRouteMode –routeWithTimingDriven true

主要是为了确保，EDI 工具在进行布线的过程中，要尽最大的努力，用最短的线长，来对两个器件进行连接。

布线完成之后，利用 EDI 工具自动进行时序优化的阶段，主要使用 optDesign 这条命令，进行包括时序优化，噪声优化等在内的各种优化。在不同阶段，optDesign 这个命令需要设置不同的选项来进行优化，比如

在 pre-CTS 阶段，使用

optDesign –preCTS；

在 CTS 和 post-CTS 阶段，使用：

optDesign –postCTS；

在布线完成之后，使用：

optDesign –postRoute

之所以在不同阶段使用不同的命令和选项，是考虑到在后端设计的不同阶段，时序优化的侧重点是有所不同的。在放置标准单元（placement）阶段，由于尚未对时钟树进行综合，标准单元的互连线尚未绕好，EDI 工具在进行时序优化的时候主要是考虑把相关的标准单元按照时序的相关要求（主要从 floor plan 和 SDC 约束考虑），放在 EDI 工具认为的最能满足时序要求的区域；在时钟树综合完成之后，侧重于优化与时钟树相关的时序问题，比如说平衡不同时钟之间的 skew；减小或者增大某些时钟的 latency；改变时钟树综合过程中使用到的一些驱动能力太弱或者 cell delay 值太大的 cell；检查 clock net 是否存在太大的线延迟，从而需要在线的中间插入相应的 buffer 或者 inverter 来增强 cell 的驱动能力，减小线延迟等等。

前面已经提到过，在进行布线之前，EDI 工具默认只会修复 setup 时序违例，而不会对 hold 时序违例进行优化。在布线之后，我们通常使用下面一系列的命令进行 setup 和 hold 的时序优化：

```
optDesign –postRoute
optDesign –postRoute –si
optDesign –postRoute –hold
optDesign –postRoute –hold –si
```

具体的流程为：以上 4 条命令依次执行，分别修复 setup 时序违例；修复因为前一步优化所引起的噪声违例；修复 hold 时序违例；再次修复因为修复 hold 违例所引起的噪声违例。以上步骤视情况可以进行多轮，但从实验的结果来看，在第二轮优化的过程中，EDI 工具实际上已经不能再继续修复大量的 setup 和 hold 时序违例。在 postRoute 阶段，EDI 工具优化 setup 和 hold 时序违例之后，如果相应的时序违例数目依然很庞大，我们是不能寄希望于让 EDI 工具继续自动进行优化。造成这种情况的原因是多方面的，最重要的原因还是在于布图规划没有做好。我们必须考虑重新调整 floorplan 或者重新进行时钟树综合。只有在 EDI 工具优化完时序之后，结果在可控制范围之后，才能进行相应的手工 ECO。如果时序违例过多，全部要通过手工 ECO 来修复的话，不仅工作量过大，耗费的时间也太多，会大大增加整个芯片的设计周期。

在修复 setup 时序违例的过程中，主要采用以下方法：

1)首先分析 data path，检查路径上是否存在某些器件的线延迟或者器件本身的延迟很大。

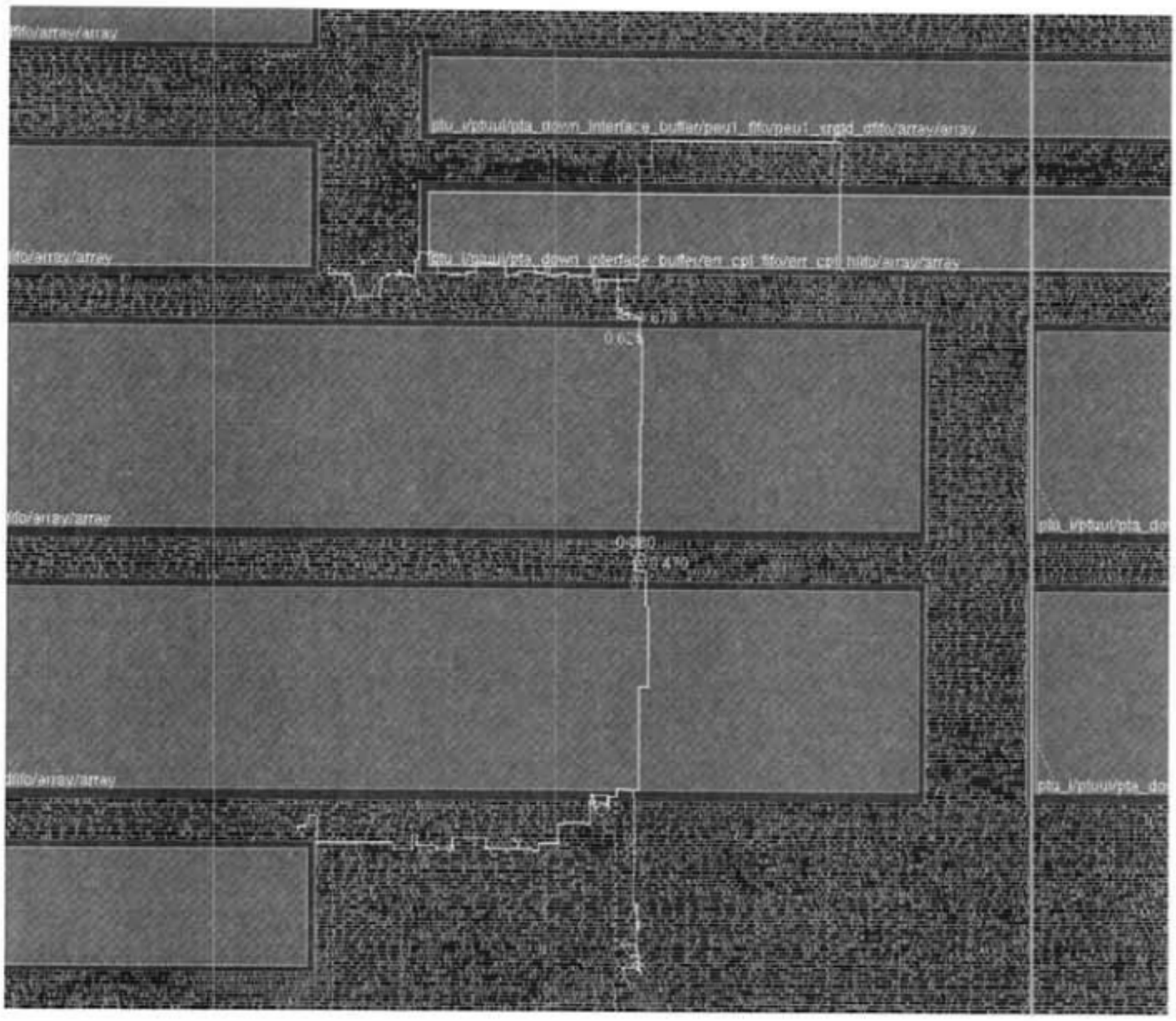
通常引起这种现象的原因是由于器件的驱动能力太弱，可以通过增大器件的尺寸来增强其驱动能力，以减少线延迟和器件本身的延迟。图 4-3 所示路径就属于这种情况。

serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC445_pcs_tx7_data_7/I (INVD1BWP12THVT)		0.022 *	2.953 f
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC445_pcs_tx7_data_7/ZN (INVD1BWP12THVT)		0.076 *	3.028 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFN445_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFCC12199_FE_OFN445_pcs_tx7_data_7/I (BUFFXD2BWP12TLVT)		0.000 *	3.028 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFCC12199_FE_OFN445_pcs_tx7_data_7/Z (BUFFXD2BWP12TLVT)		0.051 *	3.079 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFCN12199_FE_OFN445_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11191_FE_OFCN12199_FE_OFN445_pcs_tx7_data_7/I (BUFFXD2BWP12TLVT)		0.002 *	3.081 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11191_FE_OFCN12199_FE_OFN445_pcs_tx7_data_7/Z (BUFFXD2BWP12TLVT)		0.056 *	3.137 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFN11191_FE_OFCN12199_FE_OFN445_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11192_FE_OFCN12199_FE_OFN445_pcs_tx7_data_7/I (BUFFD1BWP12TLVT)		0.018 *	3.154 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11192_FE_OFCN12199_FE_OFN445_pcs_tx7_data_7/Z (BUFFD1BWP12TLVT)		0.063 *	3.217 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFN11192_FE_OFCN12199_FE_OFN445_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC446_pcs_tx7_data_7/I (INVD2P3BWP12THVT)		0.000 *	3.217 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC446_pcs_tx7_data_7/ZN (INVD2P3BWP12THVT)		0.046 *	3.263 f
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFN446_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFCC20427_FE_OFN446_pcs_tx7_data_7/I (BUFFXD6BWP12TLVT)		0.014 *	3.277 f
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFCC20427_FE_OFN446_pcs_tx7_data_7/Z (BUFFXD6BWP12TLVT)		0.038 *	3.315 f
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFCN20886_FE_OFN446_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11189_pcs_tx7_data_7/I (INVD2P3BWP12T)		0.004 *	3.319 f
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11189_pcs_tx7_data_7/ZN (INVD2P3BWP12T)		0.039 *	3.358 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFN11189_pcs_tx7_data_7 (net)	1		
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11190_pcs_tx7_data_7/I (INVD1P25BWP12T)		0.020 *	3.378 r
serdes_x16_top/serdes_x16_top_i/pcie_phy/pcs/FE_OFC11190_pcs_tx7_data_7/ZN (INVD1P25BWP12T)		0.053 *	3.431 f

图表 5 图 4-3 时序分析实例 1

从图 4-1 中所示的时序路径可以看到，路径上存在较多的驱动能力偏弱的缓冲器，这种情况造成的建立时间违例是比较容易修复，只需要将这些小的缓冲器变为 BUFFD8BWP12T 就可以大大减少器件延迟和互连线延迟。

有些 data path 是因为路径上的某些器件的负载太多，其中一部分负载或全部负载距离器件过远，造成互线延迟很大。对于这种情况，我们通常采用把负载按照分布的区域重新进行分组，通过 buffer 或者 inverter 重新连接到这个器件上来，以此来增大这些器件的驱动能力，减少线延迟。



图表 6 图 4-4 时序分析实例 2

2）在确定 data path 已经没有办法进行修复的情况下，我们才会去考虑调整时序路径的起点（start point）和终点（endpoint）的 clock latency。即通过减小 start point 的 clock latency 或者增大 endpoint 的 clock latency 来满足 setup 时序检查的时序要求。我们之所以需要首先从 data path 入手来进行时序违例的修复，很重要的一个原因在于：

1）如果需要减小起点的 clock latency，那么以这个触发器作为终点的时序路径就有可能出现 hold 时序违例。

2）如果需要增大终点的 clock latency，那么以这个触发器作为起点的时序路径就有可能出现 hold 时序违例。

所以在进行 setup 时序修复的过程中，如果需要调整触发器的 clock latency，必须充分考虑到其是否具有足够的保持时间的余量（margin）。

从 PT 进行时序分析的特点来看，如果需要调整触发器的 clock latency，最好的方法是增大 end point 的 clock latency，而不是减小 start point 的 clock

latency。原因在于，PT 在 report timing 的时候，我们通常会使用-nworst 1 的选项，也就是说，对于同一个作为 end point 的触发器，会有不止一个触发器作为 start point，而 PT 是只会报出 slack 最大的那条路径所对应的 start point。这样一来，在我们看到的时序分析报告之中，只能看到其中那条时序最恶劣（worst）的路径。所以如果我们通过减小 start point 的 clock latency 来修复 setup 时序违例的时候，只是修复了其中的一条路径，还是会有其他的时序违例出现的，这样会大大降低我们进行时序分析的时间周期。

下面这条路径，它的 data path 已经没有足够的空间进行修复了，只能通过调整起点或终点的 clock latency 来进行修复。

Point	Fanout	Incr	Path
clock clk_io (rise edge)		0.000	0.000
clock network delay (propagated)		1.022	1.022
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array/CLKB (sadrlskkb2p64x132m1b1w0c1p1d0t0)		0.000	1.022 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array/QB[11] (sadrlskkb2p64x132m1b1w0c1p1d0t0)		0.795 *	1.817 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_QB_11 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFC68189_array_QB_11/I (BUFFD2BWP12THVT)		0.014 *	1.831 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFC68189_array_QB_11/Z (BUFFD2BWP12THVT)		0.072 *	1.903 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFN68189_array_QB_11 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFCC130316_FE_OFN68189_array_QB_11/I (BUFFXD12BWP12T)		0.000 *	1.903 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFCC130316_FE_OFN68189_array_QB_11/Z (BUFFXD12BWP12T)		0.032 *	1.935 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFCN130316_FE_OFN68189_array_QB_11 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFCC132493_FE_OFN68189_array_QB_11/I (BUFFXD16BWP12THVT)		0.005 *	1.940 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFCC132493_FE_OFN68189_array_QB_11/Z (BUFFXD16BWP12THVT)		0.037 *	1.977 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/FE_OFCN132493_FE_OFN68189_array_QB_11 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_MEM_INTERF_INST/QB_IN[11] (ptu_pipe0_clk_MBIST1_LVISION_MEM6_INTERFACE_49540_ptu)		0.000 *	1.977 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_MEM_INTERF_INST/QB_IN[11] (net)			
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_MEM_INTERF_INST/p0484A/I0 (MUX2XD2BWP12TLVT)		0.010 *	1.987 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_MEM_INTERF_INST/p0484A/Z (MUX2XD2BWP12TLVT)		0.077 *	2.065 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_MEM_INTERF_INST/QB[11] (net)	4		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/array_MEM_INTERF_INST/QB[11] (ptu_pipe0_clk_MBIST1_LVISION_MEM6_INTERFACE_49540_ptu)		0.000 *	2.065 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/QB[11] (net)			
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/array/QB[11] (ptu_wrapper_2p_64x132_s_49567_ptu)		0.000 *	2.065 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/read_data[7] (net)			
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0493A/A4 (XOR4D1BWP12TLVT)		0.025 *	2.089 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0493A/Z (XOR4D1BWP12TLVT)		0.127 *	2.217 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/n_38 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0523A/A3 (XNR4D1BWP12TLVT)		0.000 *	2.217 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0523A/ZN (XNR4D1BWP12TLVT)		0.117 *	2.333 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/n_56 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0549A/A2 (XOR2D4BWP12T)		0.010 *	2.343 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0549A/Z (XOR2D4BWP12T)		0.091 *	2.434 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/n_62 (net)	2		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0524A/B2 (OAI221D1BWP12TLVT)		0.000 *	2.434 r
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0524A/ZN (OAI221D1BWP12TLVT)		0.040 *	2.474 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/n_60 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0560A/C (AO221D4BWP12TLVT)		0.000 *	2.474 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0560A/Z (AO221D4BWP12TLVT)		0.111 *	2.585 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/n_64 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0639A/A1 (OA21XD8BWP12T)		0.023 *	2.608 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/p0639A/Z (OA21XD8BWP12T)		0.063 *	2.671 f
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/n_88 (net)	1		
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/parity_err_reg/D (SDFKCSNQD4BWP12TLVT)		0.007 *	2.678 f
data arrival time			2.678
clock clk_io (rise edge)		2.000	2.000
clock network delay (propagated)		0.835	2.835
clock reconvergence pessimism		0.019	2.854
clock uncertainty		-0.065	2.789
ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rrqth_hfifo/parity_err_reg/CP (SDFKCSNQD4BWP12TLVT)			2.789 r
library setup time		-0.118 *	2.671
data required time			2.671
data required time			2.671
data arrival time			-2.678
slack (VIOLATED)			-0.008

图表 7 图 4-5 时序分析实例 3

在修复 hold 违例的过程中，主要采取以下方法：

1）增大 data path 的到达时间，即在 data path 中插入一些延迟值比较大的 buffer 或成对的 inverter。这种方法的关键是要找准插入点。我们通常会检查这条路径上所有器件的建立时间余量，选取余量最大的点插入 buffer，以此来减少在修复 hold 时序违例的过程中可能产生的新的 setup 时序违例。

```
#-------------------------------------------------------------------
# Startpoint: ptu_i/peul/peu_clk_rst/pclk_link_down_rst_n_d2_reg/CP (SDFCND4BWP12T)
# Endpoint:   ptu_i/peul/DWC_pcie_rc/u_cx_pl/u_xmlh/u_xmlh_ltssm/ltssm_lanes_active_reg_12/SDN (SDFSND4BWP12TLVT)
# slack (VIOLATED) -0.007
#           ptu_i/peul/DWC_pcie_rc/u_cx_pl/u_xmlh/u_xmlh_ltssm/ltssm_lanes_active_reg_12/SDN (SDFSND4BWP12TLVT) 1.31
#           ptu_i/peul/DWC_pcie_rc/u_cx_pl/u_xmlh/u_xmlh_ltssm/ltssm_lanes_active_reg_12_DAVID_eco0223__BUF_i_0/I (BUFFD1BWP12THVT) 1.313
#           ptu_i/peul/DWC_pcie_rc/FE_OCP_RBC122540_FE_OFN69624_FE_OCP_RBN63616_FE_OFN45327_net1/I (INVD4BWP12TLVT) 0.972
#           ptu_i/peul/DWC_pcie_rc/FE_OCP_RBC122515_FE_OFN69624_FE_OCP_RBN63616_FE_OFN45327_net1/I (BUFFD3BWP12TLVT) 0.972
#           ptu_i/peul/DWC_pcie_rc/FE_OCP_RBC101448_FE_OFN69624_FE_OCP_RBN63616_FE_OFN45327_net1/I (BUFFD3BWP12TLVT) 0.950
#           ptu_i/peul/DWC_pcie_rc/FE_OCP_RBC121900_FE_OFN45327_net1/I (BUFFD3BWP12TLVT) 0.931
#           ptu_i/peul/FE_RC_193_0/A1 (TPAOI21D16BWP12TLVT) 0.843
#           ptu_i/peul/FE_OCP_RBC120670_FE_OFN44303_peul_rst_n/I (BUFFXD16BWP12TLVT) 0.843
#           ptu_i/peul/FE_OCP_RBC121851_FE_OFN44303_peul_rst_n/I (BUFFXD8BWP12TLVT) 0.611
#           ptu_i/peul/FE_SID_C134170_FE_OFN44303_peul_rst_n/I (BUFFXD2BWP12T) 0.611
#           ptu_i/peul/peu_clk_rst/FE_RC_17996_0/A1 (INR2D16BWP12TLVT) 0.611
#           ptu_i/peul/peu_clk_rst/FE_RC_17997_0/I (INVD1P25BWP12TLVT) 0.611
#           ptu_i/peul/peu_clk_rst/FE_RC_2086_0/A2 (ND2D1BWP12THVT) 0.631
```

图表 8 图 4-6 时序违例分析 4

2）调整 start point 或者 endpoint 的 clock latency。与修复 setup 时序违例的情况相同，我们通常只会在 data path 已经没有修复余地的时候，采取考虑调整起始点的 clock latency。即增大 start point 或者减小 endpoint 的 clock latency。

4.2.3 时序分析特例

通常情况下，在时序分析时，我们只对单个时钟周期内的各种时序路径进行分析。但在实际应用中，除了最常见的对单个时钟周期进行分析外，芯片设计中的时序分析会出现一些特例。他们包括“多周期路径”（multi cycle path）和“虚假路径”(false path)。

通常情况下，对建立时间和保持时间进行分析时，是在同一周期的时钟“起沿”（launch edge）与“捕沿”（capture edge）之间进行检查。但在实际应用中，有些 data path 很长，在一个时钟周期内，很难满足时序方面的要求，尤其是如果不同的 block 之间存在两个触发器之间存在时序检查的情况的时候，这样 data path 横穿两个 block，距离很远，在一个时钟周期内无法满足时序检查的要求，就需要设置“多周期路径”：

set_multicycle_path 2 –from {FF1/CP} –to {FF2/D}

时序分析中的另一个特例为“虚假路径”（false path）。有两种类型：“功能

性虚假路径”和“主观性虚假路径”。

所谓“功能性虚假路径”，是指某些路径根本不可能在电路中实现，在逻辑上、功能上不可能达到的路径。所谓“主观性虚假路径”是由设计者对一些很长的组合逻辑电路或者两个触发器之间的时序检查（register to register）的电路路径定义为虚假路径。

虚假路径通常由客户在 SDC 中给出定义。在后端优化的过程中，如果发现某些路径不应该存在时序检查，可以将相关路径的时序报告交给前端的工程师，由他们来决定是否需要设为“虚假路径”（false path）。设为虚假路径的时序路径是不会进行时序检查的。

4.3 本章小结

本章主要对静态时序分析的重要性以及原理做了简单的分析；重点介绍了引起静态时序违例的因素及相应的解决方法；并结合项目中遇到的实际问题，分析了如何去解决时序违例。

在完成布局布线之后，大部分的精力都会放在修复 setup 和 hold 的时序违例上，所以说，静态时序分析在整个后端设计的过程中，还是占据着非常重要的地位的。只有所有的时序路径都不存在问题之后，才能保证制造出来的芯片能够正常工作。

在后端设计的前期，主要依靠 EDA 工具对时序违例的路径进行分析和优化。在后期，为了减少时间周期，提高效率，我们在修复 setup 和 hold 的过程中，会尽可能通过写一些脚本来自动检查和修复。在静态时序分析的过程中，如何高效的达到时序收敛的效果，是需要通过在实际项目中不断学习和积累经验的。

第五章 信号完整性分析及签核

5.1 串扰的产生

随着光刻和集成电路制造工艺的不断进步，芯片的特征尺寸也从深亚微米到纳米不断减小。在工艺进步的前提下，沟道长度可以做到更小。当逻辑门的沟道长度减小时，逻辑门的开关时间会相应减小，这意味着输出驱动器的上升时间会变短，或者说时钟频率可以更高，这对于提升芯片的整体性能是很有帮助的。但与此同时，所有与信号完整性 SI(signal integrity)相关的问题都会变得更加严重。串扰产生的原因，主要是由于互连线之间的寄生耦合引起的。而随着芯片特征尺寸的不断减小，互连线的物理尺寸以及间距也随之不断变小，这样只会导致互连线之间的耦合效应不断增大，因而造成互连线之间的串扰现象更为严重。

在芯片设计中，我们是需要避免串扰的产生的。因为一方面，串扰会增加受侵害网络的线延迟，从而影响芯片的时序性能；另一方面，串扰会增加毛刺产生的几率，从而影响芯片的正常工作。通常，在对串扰进行分析时，我们习惯于将产生串扰信号源的网络称为侵害网络（aggressor net 或 attacker），将受到干扰的网络称为受害网络（victim net）。当侵害网络上的信号进行高低电平的转换时，受害网络上就会产生相应的串扰噪声，从而使受害网络的信号转换速率变快或变慢，干扰甚至破坏了受害网络的正常信号传输。而毛刺的产生，是由于噪声串扰有可能使信号出现非单调转换，在这种情况下，受害网络的信号将会产生一个伪脉冲，我们称之为毛刺（glitch）。这些脉冲破坏了门电路所保存的状态，是门电路不起作用，或起错误作用，从而引起带有门电路输入的触发器和锁存器出现功能障碍[27]。

5.2 串扰的预防和修复

如何解决串扰问题，是集成电路后端设计中一个非常重要的方面[28]。我们通常从预防和修复两个方面着手。主要采用的方法包括：提高受害网络驱动器的驱动能力(即通过增大驱动器的尺寸来增强其驱动能力)；在受害网络的中间插入

buffer 或者一对 inverter 来增强受害网络抵御干扰的能力；与之对应，同样可以通过降低侵害网络驱动器的驱动能力来将对其对已受害网络的干扰能力；通过增大受害网络和被侵害网络之间的间距来减少受害网络的影响。

一、 串扰的预防

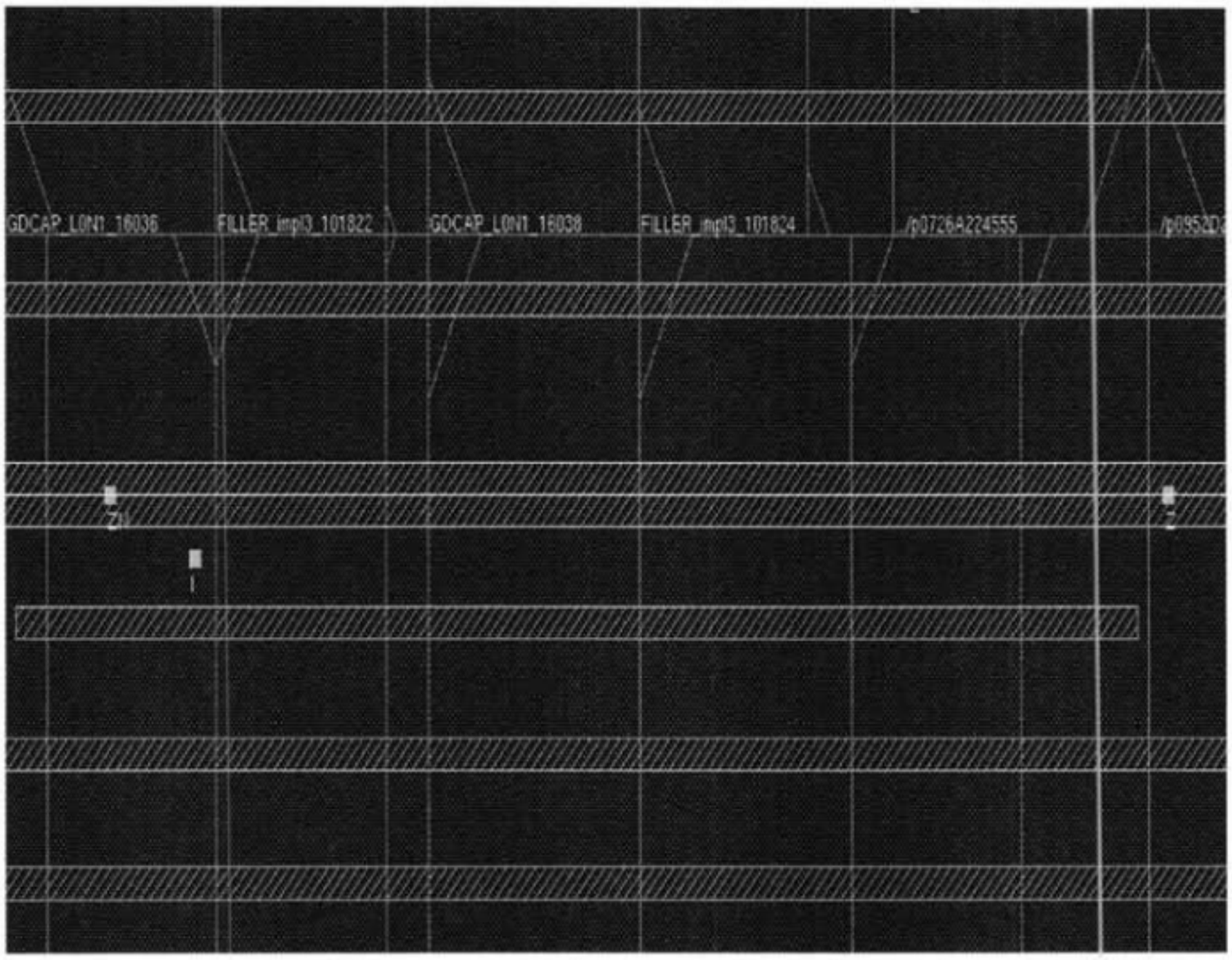
在芯片的设计的开始阶段，就需要充分考虑到如何尽最大可能去避免串扰的产生。一个良好的串扰预防措施，不仅可以降低后期修复噪声的时间，也会将串扰对芯片时序的影响降到最低。对于串扰的预防，主要采取两种方法：

1、保护受害网络

从上面的分析可以知道，串扰噪声主要产生于信号翻转的过程之中。而作为时钟控制的芯片，整个芯片的时钟网络是翻转频率最高的部分，而芯片的时钟网络对于整个芯片的时序结果又是有重大影响的。所以，为了有效的降低由时钟信号翻转所造成的串扰，我们需要合理的设计时钟网络。

随着芯片规模的不断增大，整个芯片的时钟网络也是相当庞大的。一款芯片有可能包含多种不同时钟频率的时钟网络。而每一个时钟网络，是需要驱动数以万记的组合逻辑电路和触发器。时钟树网络中的每一个接收器都可能引起一个小的串扰增量[29]，这些由时钟源至目的地的延迟总增量叠加之后，足以引起一个很大的串扰。为了避免这种情况的发生，对于始终网络上的走线，我们会在绕线之前，将这些时钟网络上的走线设定为双倍宽度、双倍间距（double width double spacing），甚至三倍宽度、三倍间距；并且时钟网络上的走线（称之为 clock net）会在时钟树综合是就完成绕线，优先于其他的非 clock net。这样能最大程度的满足我们所设定的双倍宽度、双倍间距的条件。某些特殊条件下，还可以考虑将时钟网络布线限制在拥挤程度不高的顶层金属层内，可有效消除时钟网络的串扰延迟[30]。

图 5-1 就是设定双倍宽度、双倍间距之后的效果图：



图表 9 图 5-1 双倍宽度、双倍间距效果图

2、增强受害网络的抗串扰能力

所谓增强受害网络的抗串扰能力，就是在容易受到串扰的路径上，应尽量使用驱动能力较强的器件。也正是基于这个方面的考虑，在时钟树综合阶段，我们才会对用于进行时钟树综合的 clock buffer 限定为以下几种类型：CKBD4BWP12T CKND12BWP12T CKND16BWP12T。在设计中，我们会禁止使用 TSMC40nm 单元库中驱动能力为 D0 的器件，同时，我们也会将客户给的初始网表中所有的 D0 大小的器件全部替换为 D1 或 D2 大小的器件，一个很重要的原因就是这种 D0 的器件驱动能力太弱，容易受到串扰的影响（另一个重要的原因在于这种器件驱动能力弱，或造成很大的线延迟，影响时序性能）。

二、 串扰的修复

Cadence 公司的 ETS（Encounter Timing System）可以对噪声进行分析。串扰的修复方法，总的来说，可以归结为三个方面：

1、增强驱动器的驱动能力

通过增强受害网络上驱动单元的驱动能力，可以在一定程度上减小侵害网络对受害网络的串扰影响，如果串扰不是很多，仅仅依靠增强驱动器的驱动能力，

也是有可能完全修复串扰的，但同时我们需要考虑到，驱动器的驱动能力也不能无限制的增大，需要根据噪声分析报告来决定需要把驱动器的驱动能力增强到什么程度。如果受害网络上的驱动单元驱动能力过强，就有可能称为新的侵害网络，造成新的噪声串扰的产生。

图 5-2 就是 ETS 产生的噪声分析报告的一部分：

```
Report Options:
-------------------------------------------------------------------------------
Slack     : yes
Sort by   : noise (receiver input peak)
Threshold : 324.0 (mV)
Level     : VH
-------------------------------------------------------------------------------
Peak(mV)  Level  TotalArea  %AreaTillPeak  Width(ps) VictimNet
 382.333     VH     30.93    21.23         161.80  ptu_i/ptuul/pta_down_interface_buffer/peul_fifo/peul_rcplh_hfifo/FE_OFC66775_peu.cpl_rhfifo_wdata_62:I {
ptu_i/peulcpl_rhfifo_wdata_62}

Receiver output peak:
Value         ReceiverNet
4.194 (648.000)  ptu_i/ptuul/pta_down_interface_buffer/peul_fifo/peul_rcplh_hfifo/FE_OFC66775_peulcpl_rhfifo_wdata_62/I (BUFFXD2BWP12')

Constituents:
Source     Peak(mV)  Offset(ps)  Slew(ps) Xcap(fF)  Edge  Net     TraceBackNet(NoiseType)
Cpl:       172.482    4972.000    25.000   11.008     F  ptu_i/ptu_pipe0_clk_MBIST1_MBIST_I2/MBIST_CTL_COMP/GO_ID_REG_33 -
Cpl:       128.469    4960.000    25.000   10.591     F  ptu_i/ptuul/pta_down_interface_buffer/peu0_fifo/peu0_rcpld_dfifo/array/array_MEM_INTERF_INST_DA_9 -
Cpl:        54.383    4972.963    25.000    4.518     F  ~ptu_i/peulcpl_rhfifo_wdata_62 -
Cpl:        26.998    4953.000    25.000    2.718     F  ptu_i/peulcpl_rhfifo_wdata_43 -
Baselevel:   0.002           -           -         -  -  -
```

图表 10 图 5-2 噪声分析报告

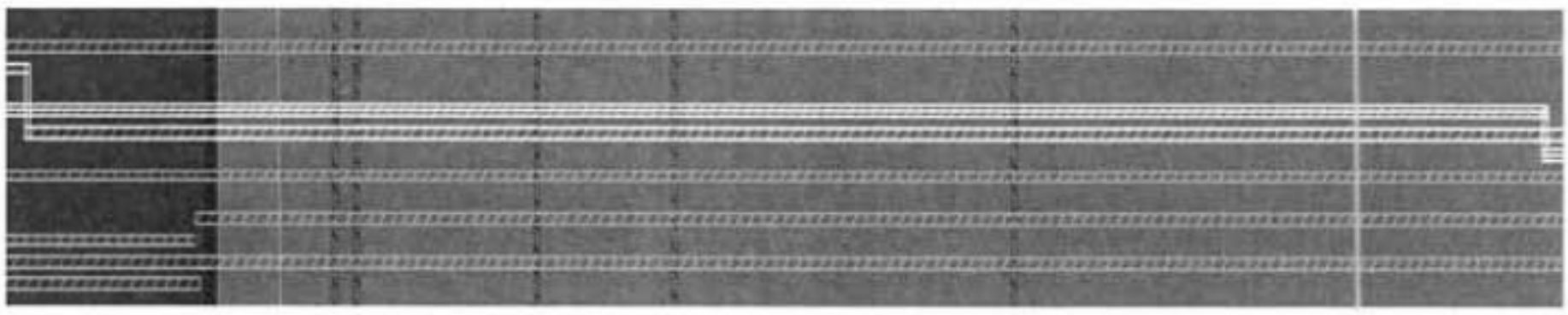
2、在受害网络上插入缓冲器

插入缓冲单元是一种有效的修复串扰的技术。很多串扰的产生，是由于两个器件之间距离过远，驱动单元不足以驱动如此长的一条线，造成线延迟很大，同时，由于走线过长，更容易受到与其并行的其他走线的影响，造成串扰增量的叠加，从而产生较大的串扰增量。不过在使用此种方法进行串扰修复时，插入缓冲器的数量不宜过多，一方面，插入过多的缓冲器，会对整条路径的建立时间造成影响，尤其是当需要在时钟网络相关的受害网络上插入缓冲器时，必然会影响到连接在这条受害网络上的触发器的 clock latency，从而对 setup 和 hold 都会造成一定程度的影响。另外，插入过多的缓冲器，同样有可能是当前的受害网络称为侵害网络。

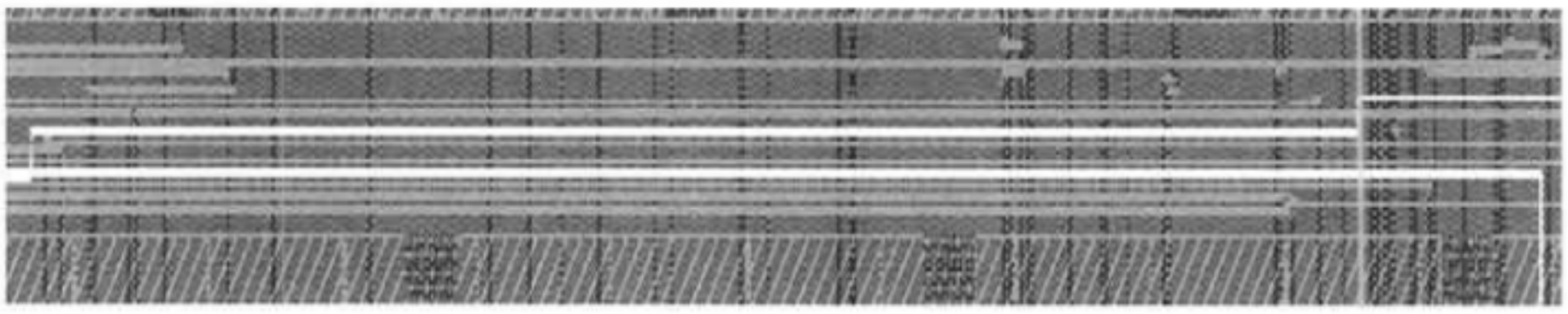
3、手工调整布线

通过手工绕线的方法，就受害网络远离主要的侵害网络，减少串扰的影响。

图 5-3、5-4 即是手工绕线前后的对比：



图表 11 图 5-3 修复串扰之前



图表 12 图 5-4 修复串扰之后

5.3 40nm 下的特殊要求

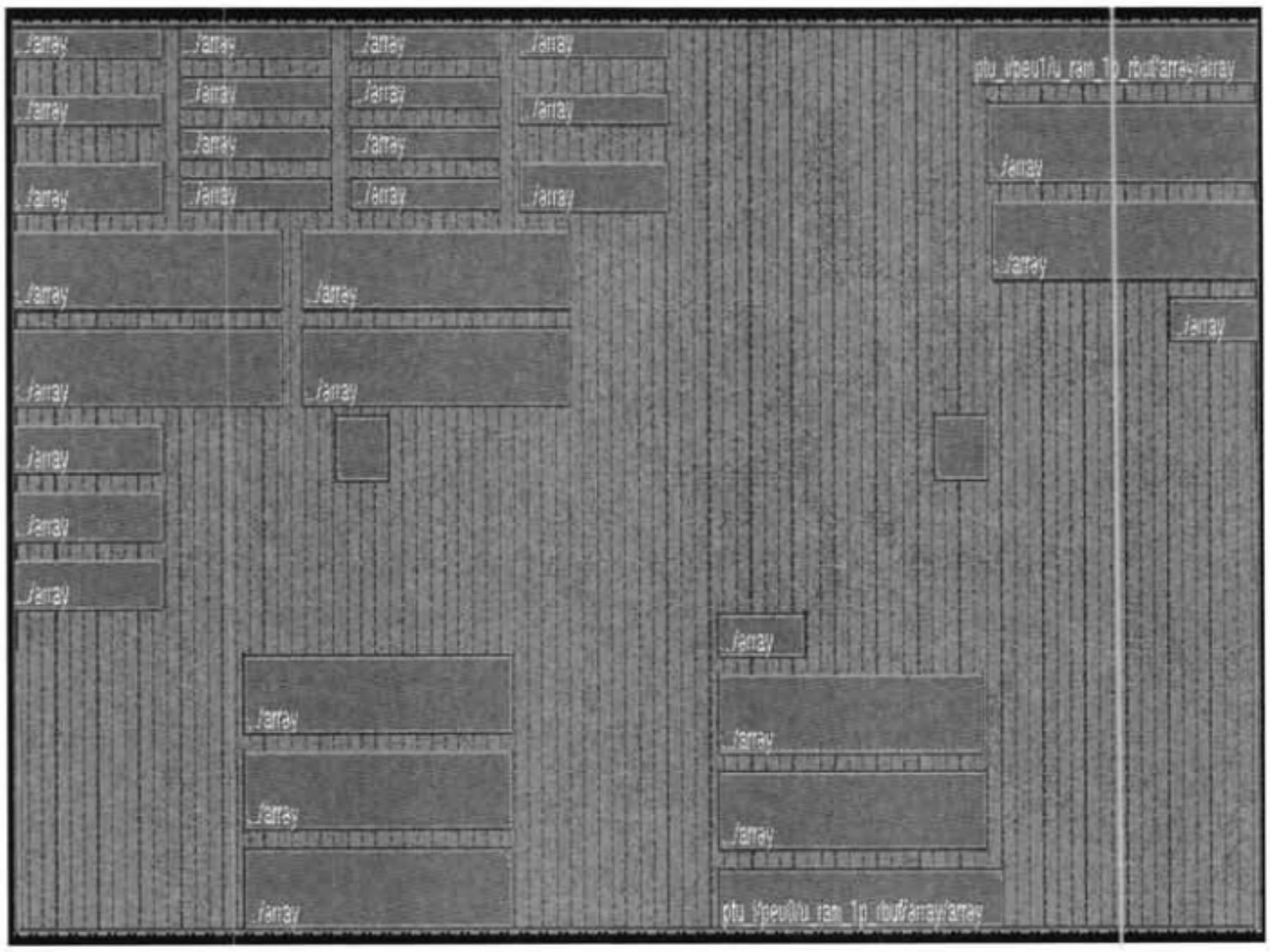
与 65nm 不同，在 40nm 工艺条件下，需要遵循一些特殊的设计规则，总结如下：

1、如果两个标准单元之间存在间隙，那么间隙的最小宽度应为 0.14um 的 2 倍以上；

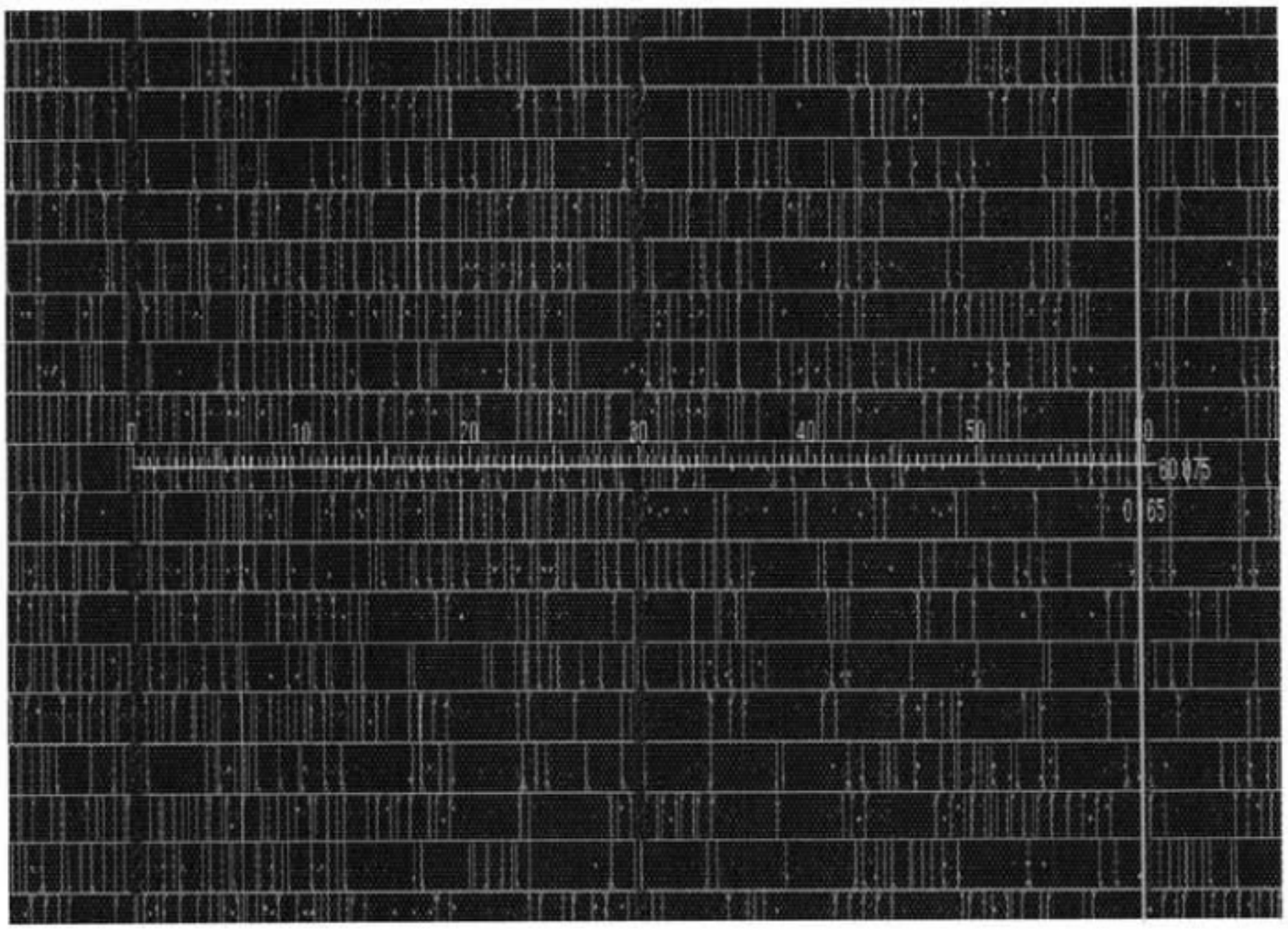
2、从整个芯片的角度，每 1500umX1500um 的范围之内，至少需要放置一个 PCLAMP 单元，并且此 PCLAMP 单元之上是不能走线的；

3、TAPCELL 的添加。规则要求，每个标准单元到 TAPCELL 的距离不能够大于 30um。

图 5-5 即是添加 TAPCELL 之后的效果图（红色标记的标准单元）：



图表 13 图 5-5 插入 TAPCELL 效果图



图表 14 图 5-6 TAPCELL 的间距

另外，为了优化时序，降低串扰，我们还采取了一下措施：

1、为了尽可能的减少不同 BLOCK 之间串扰的产生，在每个 BLOCK 的边界处需要添加宽度约为 5um 的 placement blocage 和 routing blockage。之所以需要加入 routing blockage，是因为从整个芯片的布线来看，两个相距很近的 block 之间如果存在并行的长线，极有可能造成 block 之间串扰噪声的产生，为了避免这种情况的出现，在两个 BLOCK 的边界处各加入 5um 宽的 routing blockage，这样即使两个 BLOCK 之间的通道内存在长的走线，但由于距离 BLOCK 内部的走线的间距很大，是不会造成串扰的产生。而之所以在 routing blockage 的下方加入同等宽度的 placement blockage，是因为 routing blockage 的存在会造成它下方的标准单元出现绕线的问题。

2、在 BLOCK 边界 I/0 附近，每个 I/O 都要连接一个驱动能力中等的缓冲器。这样做的目的主要在于，各个 BLOCK 在进行时序优化的过程中，只能够考虑到自己 BLOCK 内部的时序结果，但有个 BLCOK 之间是存在时序路径的，而这些路径是通过这些输入/输出端口连接在一起的，由于连个 BLOCK 之间的距离有可能较远，这样就会造成两个 BLOCK 的输入/输出端口之间的走线过长，会产生较大的线延迟，并且容易产生串扰。通过在每个输入输出端口附近插入一个驱动能力较强的缓冲器，就可以有效的解决这个问题。

5.4 签核技术

在芯片出片（tape out）之前，需要做相应的签核检查，主要包括以下几个方面：

1、时序验证：

由于现在的芯片设计采用的都是 Multi Mode Multi Corner 的方式，所以在进行时序检查的时候，需要保证所有 mode，所有 Corner 下，所有的时序路径都要满足设计要求，不能存在时序违例。同时，同样需要保证所有 mode，所有 Corner 下，noise、slew rate、max capacitance 都能够满足设计要求。

2、物理验证：

包括设计规则检查（DRC）、电路检查（LVS）、形式验证（Formality Verification）。

设计规则检查的一个主要目的在于发现与几何尺寸相关的 DRC 错误。另一

方面，由于现在硅片打磨中采用了 CMP 工艺方法，为了保持芯片每层金属密度的均匀性，需要在某些金属密度过低的地方填充金属（dummy metal），而设计规则检查能够指出哪些部分金属密度过低，需要额外填充金属。除此之外，最小面积规则检查也是设计规则检查中不可或缺的一个方面。所谓最小面积检查，主要是由于在芯片设计工艺中用到了通孔堆砌（via stacking）技术，为了保证接触的可靠性，我们需要保证通孔能够遵守最小面积规则。为了避免设计中存在最小面积的 DRC 错误，通常在布线的过程中，我们会尽可能使用双通孔（double via）技术，表 5-1 所示即为此 BLOCK 内部单通孔和双通孔的比例：

表格 5 表 5-1 单/双通孔比率

```
#Total number of vias = 4670671
#Total number of multi-cut vias = 3323544 ( 71.2%)
#Total number of single cut vias = 1347127 ( 28.8%)
#Up-Via Summary (total 4670671):
#                    single-cut          multi-cut       Total
#--------------------------------------------------------------
#  Metal 1       765765 ( 56.9%)    579038 ( 43.1%)    1344803
#  Metal 2       278423 ( 19.0%)   1185931 ( 81.0%)    1464354
#  Metal 3       161724 ( 16.0%)    846245 ( 84.0%)    1007969
#  Metal 4        88850 ( 18.0%)    405723 ( 82.0%)     494573
#  Metal 5        41035 ( 15.2%)    228922 ( 84.8%)     269957
#  Metal 6        11330 ( 12.7%)     77685 ( 87.3%)      89015
#--------------------------------------------------------------
#             1347127 ( 28.8%)   3323544 ( 71.2%)    4670671
```

电路检查的主要任务是 LVS，通常是要求先对芯片顶层的布线进行基本检查，包括是否有短路、开路，再将底层的标准单元 GDSII 数据结合在一起进行 LVS 检查。通常出现 LVS 不能通过的原因在于设计总存在短路和开路。

形式验证也称为逻辑等效验证或者逻辑等效检查（logic equivalency check）。之所以需要做形式验证，主要在于在手工 ECO 的过程中，设计者有可能将某些器件替换为与之逻辑并不对等的器件，由此造成形式验证错误的出现。在常见的错误就是在时序路径中仅仅插入了一个 inverter 缓冲器，而不是一对。

3、其他验证

为了保证芯片能够顺利制造出来，在流片之前，还会对以下几个方面进行相应的检查：

1）是否进行了通孔的优化。前面已经提到过，为了避免出现最小面积的 DRC

错误，我们会尽可能多的使用双通孔（double via）。

2）是否进行功耗的优化。具体说来就是，为了尽量较小芯片的功耗，在选择标准单元的时候，我们应该尽可能多的使用高阈值电压（HVT）的单元，因为高阈值电压的单元具有较小的泄漏电流，对整个芯片功耗的降低有很大作用。在此 BLOCK 中，使用的各种类型的标准单元的比例和数目如下所示：

HVT CELL: 176433 (43.3%)

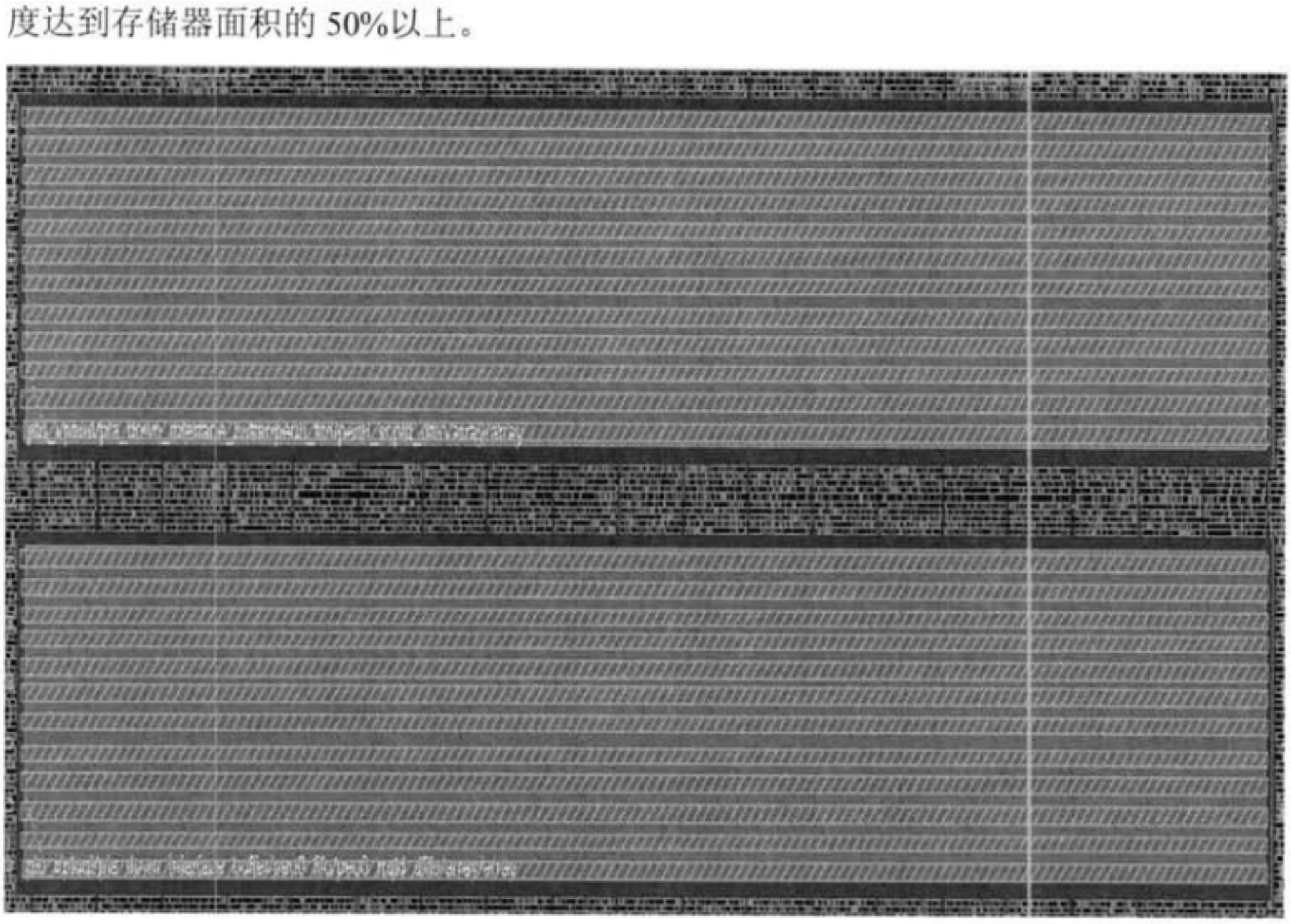
RVT CELL: 69577 (17.08%)

LVT CELL: 161425 (39.62%)

3）时钟网络中所有的标准单元都应该使用以 CK 开头的标准单元。在 TSMC40nm 的标准单元库中，有专门的用于时钟网络的标准单元，有更好的驱动能力和抗干扰能力，对于保持时钟网络的稳定性和可靠性有很大作用。

4）保证时钟网络中所有的走线都采用了双倍宽度、双倍间距的属性。此举同样是为了保证时钟网络的稳定性和可靠性。

5）对于频率大于 100MHZ 的存储器，必须保证其上面覆盖的电源网络的密度达到存储器面积的 50%以上。



图表 15 图 5-7 电源线密度效果图

6）所有的连接 1'b0 的端口都需要连接到一个 TIEL 单元之上；所有的连接 1'b1 的端口都需要连接到一个 TIEH 单元之上。不允许有悬空的端口存在。

7）芯片中应尽量避免使用驱动能力为 D0 的单元，尤其是对于处于时钟网络中的标准单元。主要是考虑到芯片工作环境的复杂性，驱动能力太弱的标准单元更容易受到外界温度、电压等的影响，这对于芯片的正常工作是一个比较大的隐患。

8）检查电源网络是否合理，也就是需要检查是否所有的标准单元的电源端口已经正确的连接到电源网络上。

5.5 本章小结

本章介绍了串扰产生的原因，并提出如何预防串扰和修复串扰的方法。在此项目中，在时钟树综合的过程中，我们已经将所有的 clock net 设定为：double width double spacing，以此来避免串扰对 clock net 的影响。而在修复串扰的过程中，如果数量过多，每一条串扰都进行仔细分析，必然会增加修复串扰的时间，降低效率，通常的做法是，报出所有被侵害网络的驱动器，增大这些驱动器单元的驱动能力。如果对一些被侵害网络，仅仅依靠增大驱动器的驱动能力，并不能完全解决串扰的问题，调整布线是一个非常有效的方法，但花费的时间会更多一些。

总之，要解决串扰问题，需要在预防和修复两方面都采取有效的措施。

另外，在 40nm 工艺条件下，存在一些不同于以往工艺的设计规则，这也是我们需要格外注意的。

在芯片 Tape Out 之前，我们必须保证所有的时序检查、设计规则检查都能够满足 foundry 厂所规定的制造要求，只有这要，制造出来的芯片才能够正常、稳定、功能正确的工作。

第六章 总结与展望

6.1 回顾和总结

随着集成电路特征尺寸的不断缩小，集成电路的功耗、面积都将面临越来越大的挑战。如何运用新的技术去解决 40nm 甚至以下的设计难题，是我们需要去学习和探讨的。同时，技术的进步使得电子产品的更新速度越来越快，而其中芯片的设计公司所面临着成本、及时上市的压力，怎样更好更快的完成芯片的设计，也是我们需要去积累和摸索的。

本文基于 TSMC 40nm 工艺的一款芯片的后端设计为例，结合 Cadence 公司的 EDI(Encounter Digital Impletation)、ETS(Encounter Timing System)， Synopsys 公司的 Prime Time，QRC 等 EDA 工具，研究了 IC 后端设计中的一些设计定制流程的方法，利用 PERL 和 TCL 等辅助语言提高后端设计的效率，缩短开发的周期，并且可重复利用。同时，针对 40nm 工艺所存在的高集成度、层次化设计、泄漏功耗、多角落一多模式（Multi-Mode-Multi-Corner）、串扰噪声等等问题，探讨和研究解决这些问题的先进技术和方法。

本文第一章主要介绍了集成电路后端技术的发展现状和前景，对本课题的来源及本人在项目中所扮演的角色都做了介绍；指出了论文的组织结构。

第二章主要是介绍在 40nm 工艺条件下，在此款芯片的后端设计中，我们所使用的基于 Encounter 的后端设计的流程，并对此流程的每一个步骤都做了简单的介绍，指出在每个步骤，EDI 工作所完成的主要工作，以及哪些方面需要特别注意。

第三章主要介绍了在布图规划的过程中所需要考虑的一些重要因素，强调了布图规划在整个后端设计的流程中所扮演的基础但却非常重要的特性。

第四章是本文的重点所在，主要介绍了静态时序分析的相关理论，着重介绍和分析了如何去解决整个设计中所遇到的时序违例的情况，并比较了再此流程中的每一阶段，时序分析优化的区别。

第五章介绍了如何去预防和消除噪声干扰，并简单介绍了，在芯片 tape-out 之前，为保证能够顺利出片，我们需要做哪些方面的检查和验证。

第六章则对整篇文章进行了总结，并指出在项目实施过程中，遇到了哪些困难、积累了哪些经验、哪些方面需要改进和提高。

6.2 下一步工作的展望

此项目是本人第一次参与基于 40nm 工艺的集成电路后端设计。40nm 工艺条件下，相对于 65nm 工艺，时序更加不容易收敛，设计规则方面的要求也更多，需要阅读很多有关 40nm 芯片设计规则及注意事项的文档，并要将文档中相应的要求，通过 EDI 工具能够识别的命令的方式，将这些要求体现在整个设计之中。比如，在 40nm 工艺条件下，对两个标准单元的间隙有了严格的规定，即不能够小于 0.28um；需要在每个 block 的内部插入 TAP CELL，并且规定每个标准单元附近 30um 的距离内，必须存在至少一个 TAP CELL；对于整个芯片来讲，需要保证在每 1500um 的范围内，至少存在一个 PCLAMP CELL 等等。

总结这个项目，有以下几方面需要提高：

1）对 40nm 条件下所规定的必须要遵守的一些设计规则，虽然知道怎么去满足这些设计规则，但对于为什么要规定这些设计规则，具体的原理和原因还不是很清楚，需要自己继续查找文档，分析原因，在提高自己动手能力的同时，也要提高自己的理论水平；

2）提高自己编写 TCL 和 perl 脚本的水平，是自己在今后的后端设计过程中，能够更多的实现自动化、脚本化，以此来提高效率；

3)当前的这套基于 40nm 的后端设计的流程，以完全不能使用于 28nm 芯片的设计，公司正在研究 28nm 工艺条件下后端设计的流程，自己也应该多学习，多研究这方面的内容。

参考文献

[1] 迟雪峰。65nm 下的 TD-SCDMA 芯片低功耗后端设计。[学位论文]复旦大学，2008.
[2] David A, Hodges, et al. Analysis and Design of Digital Integrated Circuits In-Deep Submicron Technology. 电子工业出版社,2005.
[3] Moore G E. Cramming More components onto integrated circuits[J].Electronics Magazine, 1965, 38(4):114
[4] Himanshu Bhatnagar. Advanced ASIC Chip Synthesis Using Synopsys Design Compiler Physical Compiler And Primetime. New York. USA. Kluwer Academic Publishers.20
[5] Hubert Kaeslin. Digital Integrated Circuit Design From VLSI Architectures to CMOS Fabrication. London. UB. Cambridge University Press 2008
[6] 何小虎，胡庆生，肖洁。深亚微米下 ASIC 后端设计及实例。中国集成电路。2006 年第 87 期
[7] Cadence. Encounter TM User Guide. Cadence Reference Book. 2004,06:1-100
[8] Cadence. First Encounter Ultra Lab Manual. Cadence Reference Book,2004,06:1-100
[9] Cadence. Encounter TM Menu Reference. Cadence Reference Book. 2004,06:50-51
[10] Cadence. Low-Power Methodology Kit User's Guide. Kit Version 8.2 2008, 34-3
[11] Cadence. NanoRoute TM Ultra. Cadence Reference Book. 2005,10:1-100
[12] Synopsys. PrimeTime Workshop. Synopsys Education and Training Services, 2001,06:1-100
[13] 范奉艳。基于 51 核的 SOC 物理设计与验证。[学位论文]山东大学，2008
[14] Synopsys Manual Release U-2003.06-QA,June 2000
[15] 李建军。CMOS SENSOR 接口的 ASIC 电路设计。[学位论文]上海交通大学，2008
[16] Synopsys. Chip Synthesis Workshop. Synopsys Customer Education. 2002,10:1-10
[17] Cadence. NanoRoute TM Technology Reference. Cadence Reference Book. 2004,06:57-6
[18] Cadence. NanoRoute TM Ultra. Cadence Reference Book. 2005,10:60-70
[19] Cadence. CeltIC TM User Guide. Cadence Reference Book, 2004, 06:1-10
[20] 张文耀。高性能中断逻辑部件的优化设计与物理实现。[学位论文]国防科学技术大学，2010
[21]李磊. 深亚微米高性能数字 ASIC 芯片的后端设计[硕士学位论文]：电子科技大学：2007
[22] 徐靖。X 微处理器的半定制/全定制混合设计研究。[学位论文]国防科学技术大学，2009
[23] 徐庆光。600MHz YHFT-DX 算术逻辑部件的设计与实现。[学位论文]国防科学技术大学，2010
[24] Allan A, Edenfeld D, Joyner W H Jr, et al. 2001 technology roadmap for semiconductors[J]. IEEE Trans Computer, 2002, 35(1):42-5
[25] Circuits, IEEE Journal of, Volume: 29, Issue: 6, June 1994 Pages:663 -- 6
[26] 刘凯峰。600MHz YHFT-DX 处理器物理设计。[学位论文]国防科学技术大学，2010
[27] 刘晓宇。YHFT-DX 芯片关键模块的物理设计。[学位论文]国防科学技术大学，2009
[28] Dake Liu, Svensson, C. Power consumption estimation in CMOS VLSI chips. Solid-State

[29] 骆建平。超高频无源 RFID 物理设计与低功耗时钟树综合。[学位论文]西安电子科技大学，2010
[30] 唐有情。纳米级工艺下系统级芯片的物理设计。[期刊论文]中国科技信息，2010

北京大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：李小明 日期：2012 年 6 月 3 日

学位论文使用授权说明

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

- 按照学校要求提交学位论文的印刷本和电子版本；
- 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
- 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
- 因某种特殊原因需要延迟发布学位论文电子版，授权学校□一年/□两年/□三年以后，在校园网上全文发布。

（保密论文在解密后遵守此规定）

论文作者签名：李小明 导师签名：

日期：2012 年 6 月 3 日

致谢

在此论文完成之际，衷心感谢我的导师曹喜信老师，在整个硕士研究生学习阶段对我的指导和帮助。感谢于敦山老师、程玉华老师、曹健老师，在学业、学术方面，给予的指导和帮助。在 Alchip 实习的两年当中，使我对后端设计有了更加深刻的认识，积累了丰富的后端设计经验，提升了自己在静态时序分析方面的能力。非常感谢项目经理丁贝先生对我的信任，给予我很多机会，给我提供了一个充分展示自己的舞台。感谢企业导师吴涛先生对我的指导，无论是在论文的完成过程，还是在项目的实施过程中，对于我的任何问题，都能够耐心、热心的给予解答，并教会我很多后端设计方面的技巧，使我能够更加快速的提升自己的能力。感谢我的同学兼同事张桂勇，我们共同参与完成了这个项目，感谢你在项目完成过程中对我的帮助和支持。

最后，感谢一直关心、支持、爱护我的父母和家人，是你们为我创造了如此优越的学习环境，才能让我有机会不断提升自己，创造属于自己的一片天地！