目录

摘要……..1

第一章：引言……2
1.1 ASIC 的设计流程概述……..2
1.2 综合方法……5
1.3 物理综合概述及方法简介……6

第二章：逻辑综合与 Ambit 的使用……7
2.1 逻辑综合的一些基本概念……7
2.2 Ambit 的一些特征……8
2.3 Ambit 在逻辑综合中的应用……9
2.4 设计实例……13

第三章：自动布局布线与 Silicon Ensemble 的使用……15
3.1 自动布局布线概述……15
3.2 工具 Silicon Ensemble 的介绍……16
3.3 Silicon Ensemble 在自动布局布线中的应用……17
3.4 使用 SE 的一些说明……22
3.5 设计实例……22

第四章：PKS(Physical knowledge synthesis)的使用……..24
4.1 物理综合流程的介绍……24
4.2 PKS 在 SPnR 流程中的应用……26

第五章：总结……..35

致谢 参考文献……36

摘要

本文介绍了基于标准单元库的深亚微米数字集成电路设计中的物理综合流程及其方法。此流程从设计的系统行为级描述VHDL源代码开始，依次通过设计综合，综合后仿真，到自动化布局布线。在这里，我使用Cadence公司的逻辑综合工具Ambit进行逻辑综合，用Cadence公司的自动布局布线工具Silicon Ensemble进行自动布局布线，最后介绍综合了上述两个软件功能的Cadence公司的PKS (Physical knowledge synthesis)工具在逻辑综合和自动布局布线流程中的应用。本文的第1 章主要是深亚微米数字集成电路的设计流程和其中物理综合流程的概述。从第2章开始我们将分章节详细介绍物理综合流程的主要步骤及工具使用。第2章介绍Ambit在逻辑综合中的应用，解释了综合的概念，介绍了逻辑综合的实现及讨论了几个常见问题的解决方法。第3章介绍Silicon Ensemble在自动布局布线中的应用。第4章则介绍了涵盖了Ambit和Silicon Ensemble两者功能的PKS在逻辑综合和自动布局布线流程中的应用。本文同时对各个软件在各个流程中的应用，用一个实例加以举例说明。

关键词：标准单元库， 逻辑综合， 物理综合， 自动布局布线， floorplan， Ambit，Silicon Ensemble， PKS(Physical knowledge synthesis)。

第一章　引言

1.1 ASIC 的设计流程概述

随着电路设计进入VLSI ，甚至ULSI 时代，电路规模迅速上升到几十万门甚至几百万门。根据摩尔定律，每十八个月增加一倍。而设计人员的设计能力只是一个线性增长的曲线，远远跟不上电路规模指数上升的速度。为了弥补这个差距，工业界对EDA 软件和设计方法不断提出新的要求。在80 年代，由美国国防部支持的Very High Speed Integrated Circiut 发展计划促成了VHDL 的诞生，并使之成为了国际标准。而Cadence公司的Verilog HDL 在工业界获得了广泛的接受，并最终成为了国际标准。利用HDL 进行设计大大方便了设计输入，提高了设计抽象程度，更有利于设计人员发挥聪明才智，因而可以大大提高设计效率，缩短了设计周期。

随着电路规模的增大和系统复杂度的增加，直接用电路实现已是不可能，RTL级的HDL编码也变得越来越难以忍受。行为级综合技术的发展为设计者带来了曙光。它使设计者开始逐步摆脱繁重的RTL 级编码，大大提高了设计者的设计灵活性和设计效率，减少了工艺及物理对设计的约束。

为了提高设计的速度和设计成功率，利用已验证正确的设计作为新设计的一部分是现在大规模设计的常用方法。随着时代的发展，人们对产品的要求越来越高。他们要求的不仅仅是新产品的出现，更多的是要求改善旧产品的性能，增加更多的功能。为此对旧的设计的修改是必须的。为了充分利用以前的成果，减少修改的工作量，加快设计修改速度，同时尽量不影响不变部分，提高修改的成功率，技术更改指令ECO 被提了出来并得到了发展。

随着半导体工艺的不断进步，器件的特征尺寸越来越小，线宽越来越窄，器件的速度变得越来越快。但同时随着设计的越来越复杂，电路规模的越来越大，金属线的长度和层数不断增加，线宽也随之变小。这都导致了金属连线的延时变大。于是器件的延时不再是一个系统的主要延时，连线的延时变得越来越重要，甚至超过了器件的延时。因此以前设计系统时只考虑器件延时的观念已经行不通，设计时考虑连线的延时是必须的。设计者在设计时必须同时考虑到综合和版图，且使综合和版图尽量结合在一起。把综合后的时序信息前注释到布局布线，同时布局布线后提取寄生参数和时序延时信息后注释回综合，从而使逻辑设计和物理设计紧密的结合起来。考虑到连线延时，必须进行版图后仿真。版图后仿真必须后注释大量的版图时序延时信息。

电路规模的增大导致了时钟同步的问题。时钟到达不同子模块的延时不同，这成了一个系统失败的致命弱点。为了解决时钟延时的问题，在布局布线中CLOCK TREE 的技术得到了极大的发展。它较好的解决了这时钟延时的问题。

随着系统规模的不断增大，功耗的问题变得越来越重要，散热成了人们的一大难题。为此，设计者在进行设计系统的时候必须考虑功耗的问题。在逻辑综合后必须进行功耗分析。

◆ 设计流程

基于标准单元库的数字集成电路设计方法主要流程如下：

1. 功能与规格要求。
2. 设计输入；使用硬件描述语言或者电原理图形式把设计输入到一个ASIC设计系统里。
3. 逻辑综合；使用逻辑综合工具把电路描述生成网表文件—一种逻辑单元和他们之间互连的描述。
4. 系统分片；单位电子系统的规模过大而超过一块ASIC芯片所能包含的集成度或芯片尺寸时，必须对系统进行分割，使整个网络分割成若干个小网络，其规模略小于ASIC芯片的最大尺寸，并分配到若干个集成电路的芯片上，这一设计过程称为分片。
5. 前仿真；查看设计的功能是否正确。
6. 布图规划；将网表文件中的功能块安排在ASIC的芯片上。
7. 布局设计；确定所有标准单元在功能块中的位置。其主要目的是便于或优化随后的几何布线设计，同时减少关键节点的互连延迟和芯片面积。
8. 布线设计；完成所有节点的连接。
9. 寄生参数提取；提取内部互连的电阻和电容值。
10. 后仿真；查看加了负载后设计是否还工作正常。
11. 流片

◆ 大体的流程图如图1.1 所示：

由于论文的侧重点，本文将只详细讲述几个流程，及工具在这些流程中的应用。

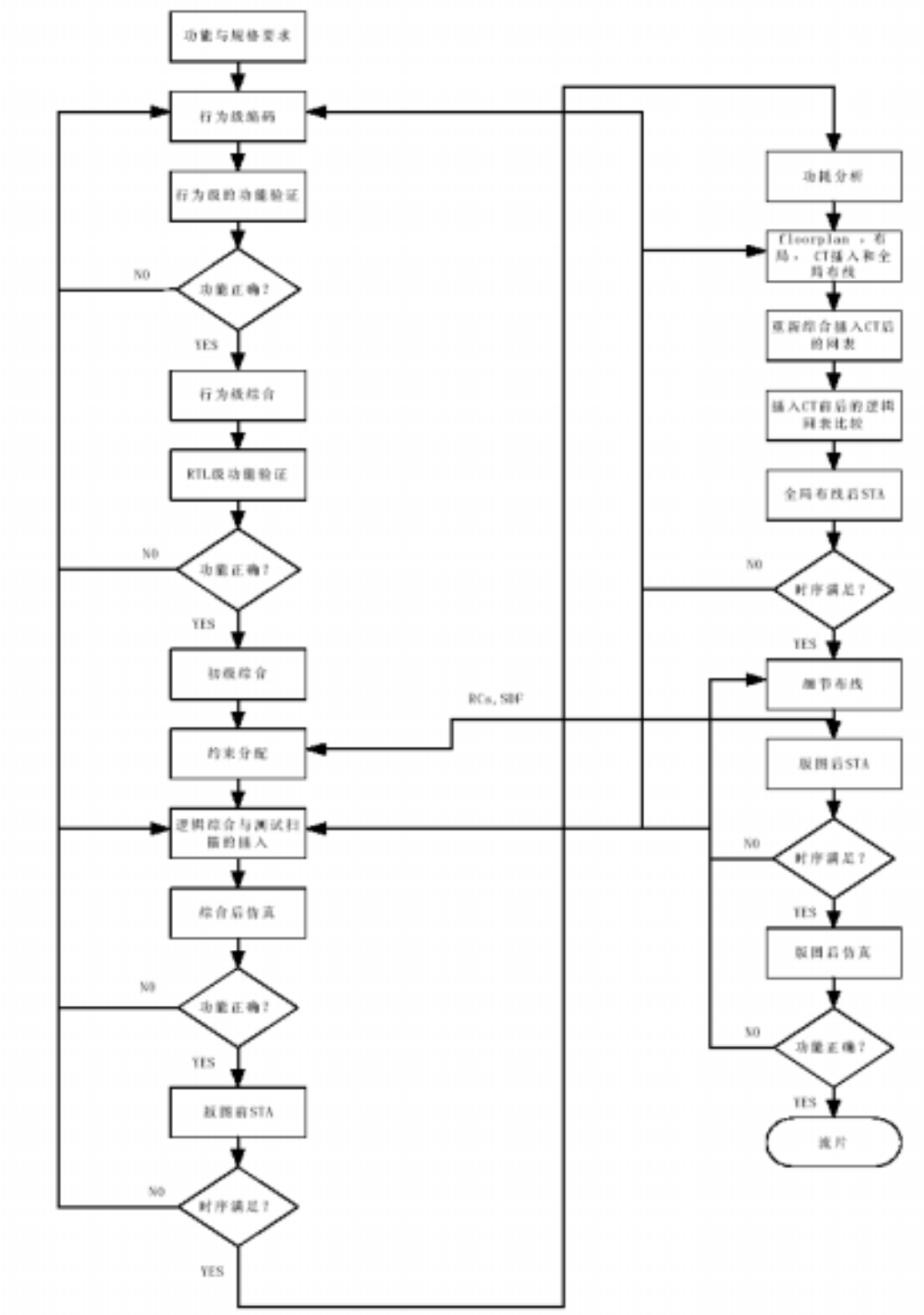


图1.1 ASIC的设计流程

1.2 综合方法

随着集成电路集成度的不断提高，对集成电路的设计提出了越来越高的要求，设计的芯片必须具备高性能，高可靠性和高保密性，同时设计周期应尽可能短。通常，综合方法是指电路从较高级别的描述自动地转换到较低级别描述地自动设计方法，即从RTL级HDL描述通过编译产生符合特定工艺地门级网表，实际上也是从行为级描述自动化生成门级电路的过程。它已成为九十年代以来ASIC设计的核心技术之一。

综合(synthesis)：就是把思想转换为实现欲想功能的可制造的设计。综合是约束驱动和基于路径的。

在这里，综合也就是把行为级或RTL 级的HDL描述转换为门级电路的过程，用公式表示就是：

综合等于= 翻译+ 优化+ 映像

(Synthesis = Translation + Optimization + Mapping)

用图形表示就是：(如图1.2所示)

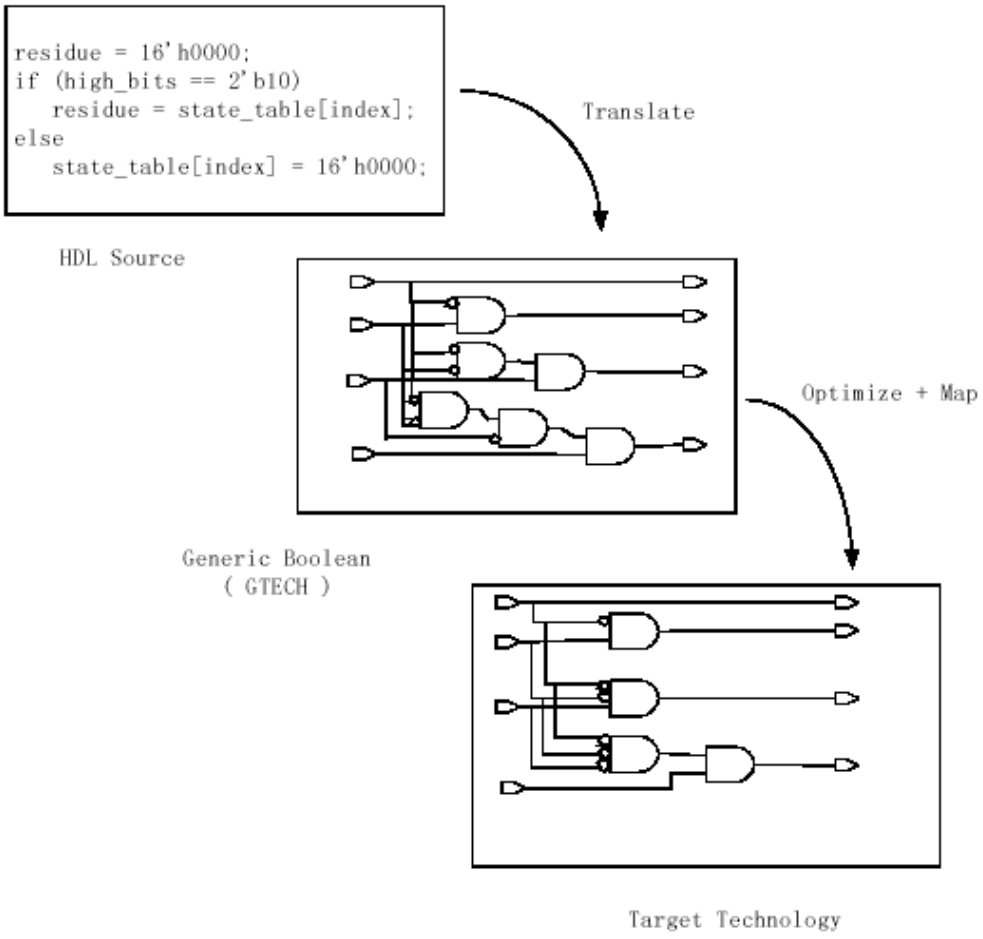


图1.2 综合的概念

对一个模型进行综合的基本步骤如图1.3所示

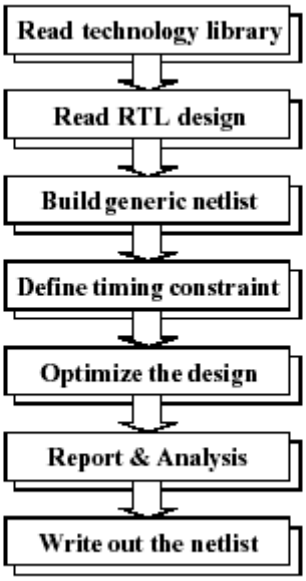


图1.3 综合的基本步骤

1.3 物理综合概述及方法简介

Synopsys公司在上海举办的2000电子设计自动化大会推出了Synopsys历经四年投巨资开发了物理级综合解决方案。本次大会把物理级综合专题作为重要的内容之一

Synopsys公司提出的“物理级综合（Physical Synthesis）”方案指的是随着半导体制造技术步入深亚微米时代，IC设计已不能以前后端来划分，而必需将逻辑设计和物理设计结合在一起。

Candence公司的PKS(Physical knowledge Synthesis)软件就是Candence公司推出的针对物理级综合解决方案的软件。该软件涵盖了整个逻辑综合和自动布局布线流程。使用PKS就可以将前端的逻辑设计和后端的物理设计结合在一起，不再截然分开的设计，实现了“物理级综合（Physical Synthesis）”。

第二章　逻辑综合与 Ambit 的使用

2.1 逻辑综合的一些基本概念

2.1.1 什么是逻辑综合？

公式表示就是：综合= 翻译+ 优化+ 映像

(Synthesis = Translation + Optimization + Mapping)

用图表示如图2.1所示

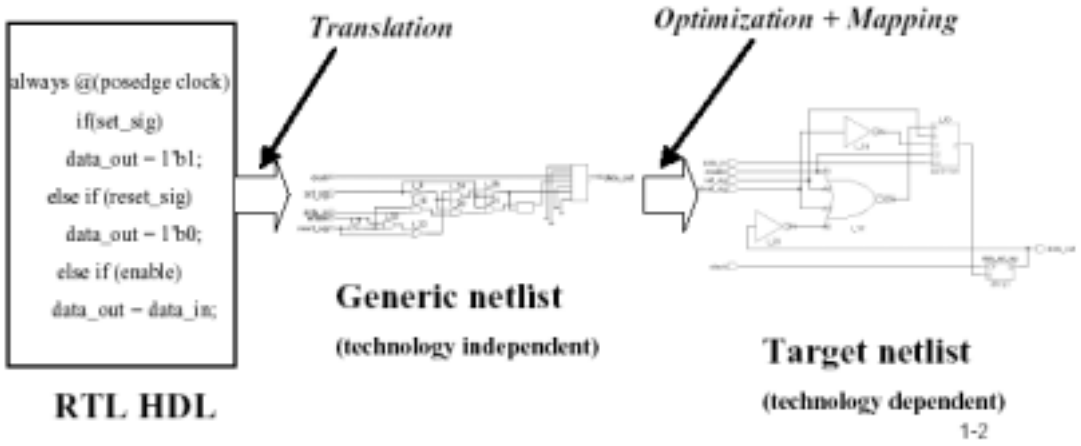


图 2.1 逻辑综合

2.1.2　Ambit 综合的特征

使用 Ambit 进行逻辑综合的特征如下图 2.2 所示：

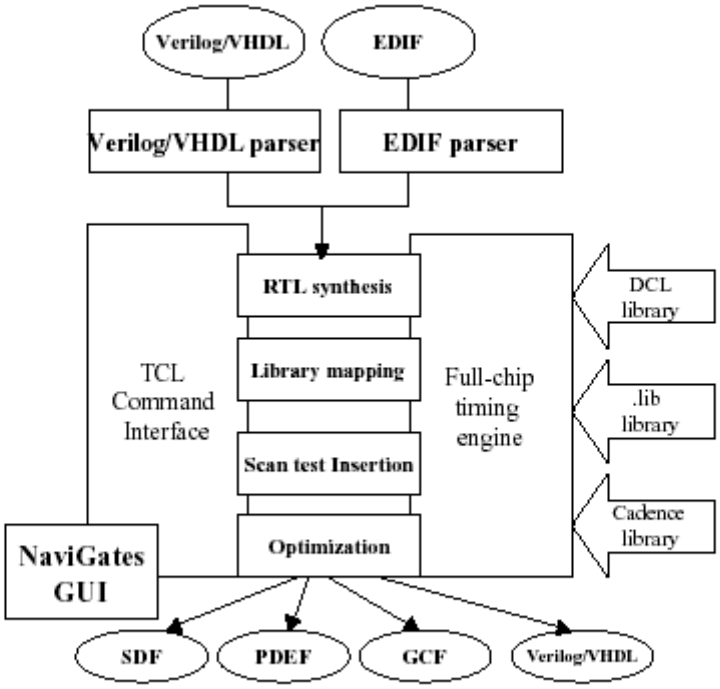


图2.2 Ambit综合特征

2.2 Ambit的一些特性

2.2.1 Ambit的一般特性

◆ 提供基于文本的TCL Shell, ac_shell。

◆ 启动Ambit综合的虚拟模式-navigates,可以使用下面两个命令：

navigates 或者 ac_shell –gui

2.2.2 navigates的界面

◆ 一个调试和分析工具，界面如图2.3所示：

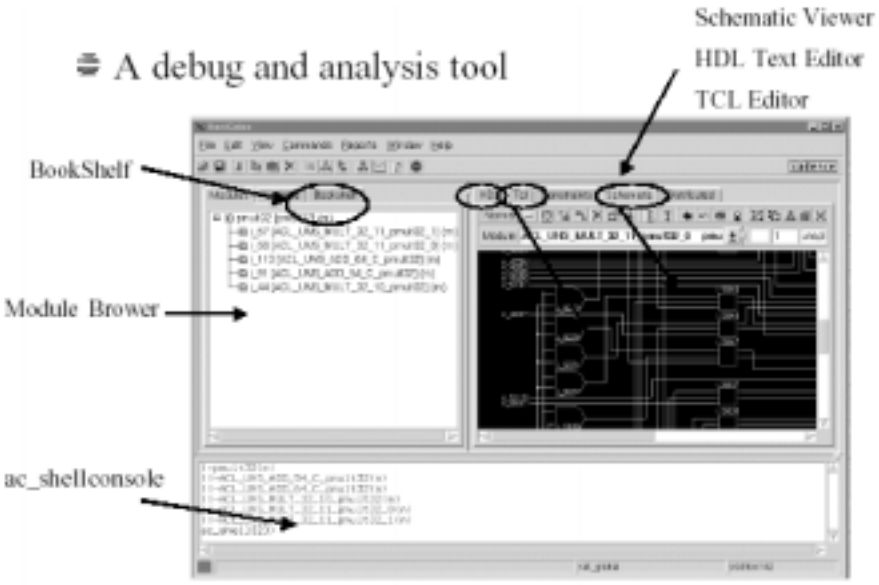


图2.3 navigates的图形界面

2.2.3 Ambit的文件管理

◆ navigates的界面提供一个完整的易理解的文件管理。

◆ 提供有HTML的帮助文件。

◆ 在ac_shell里的文本帮助文件。

在ac_shell里使用 help <command name> 命令

2.2.4 输入和输出

Ambit综合时能够

◆ 读入Verilog或者VHDL RTL门级网表文件。

◆ 输出Verilog或者VHDL门级网表文件。

◆ 读入和输出读独立于平台的Ambit Data Base(ADB)文件。

◆ 输出SDF路径约束

◆ 输入SDF标准延时文件(IOPATH/INTERCONNECT)。

◆ 使用 write_assertions 命令输出时序信息。

◆ 读入和输出PDEF文件

◆ 读入RC寄生信息

◆ 读入和输出EDIF文件

◆ 输出GCF文件

2.2.5 前端特性

◆ 分层设计
◆ 闩锁和触发推理
◆ 情况语句，有限状态机，代码选择和ambit结构提示。
◆ 移位链算法运算符和移位器的推理

2.2.6 报告的特性

◆ report_timing
◆ report_area
◆ report_hierarchy
◆ report_design_rule_violations
◆ report_library
◆ report_net
◆ report_fanin
◆ report_fanout
◆ report_fsm

通过以上命令，Ambit在综合结束后可以给出一些有关时序，面积，层次，设计规则违背，库，扇入或扇出的信息。

2.3 Ambit在逻辑综合中的应用

2.3.1 综合一个模型的基本流程

◆ 读入工艺库（设置目标库）
◆ 读入VHDL或者Verilog HDL文件
◆ 建立普通网表
◆ 定义时序约束
◆ 检查约束和网表
◆ 对目标工艺优化普通网表
◆ 分析设计并验证时序是不是达到目标
◆ 输出结果及查看相关信息

2.3.2 调用Ambit

调用Ambit有两种模式

◆ 文本模式：ac_shell
◆ 图形模式：ac_shell –gui 或者 navigates

2.3.3 读入目标库

◆ 使用下面的命令来读入工艺库

read_tlf csmc06core.tlf

注意：csmc06core.tlf（目标库）是csmc 0.6um单元库

◆ 设置你要使用的目标库

set_global target_technology csmco6core.tlf

2.3.4 读入RTL硬件描述语言

读入你要综合的RTL HDL文件

◆ 读入Verilog文件： read_verilog filename

◆ 读入VHDL文件： read_vhdl filename

2.3.5 建立普通网表

◆ 独立于工艺的网表文件必须在读入RTL文件后建立

◆ 建立普通网表的命令是：

do_build_generic

或者 do_bu (简写形式)

建立普通网表的图形界面如图2.4所示：

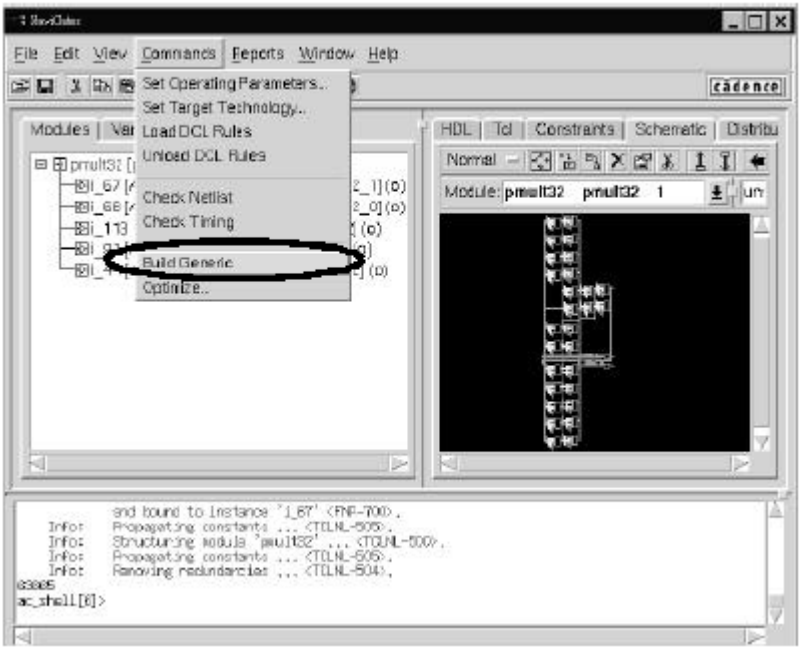


图2.4 建立普通网表

◆ do_build_generic命令

◇ 产生Control Data Flow Graphs控制数据流图表（CDFGs）

◇ 进行很好程度的优化

◇ 进行资源分配

◇ 对所有在源文件中定义的模块产生分级的网表。（使用ATL单元和ACL功能块）

2.3.6 设定约束

◆ 对Ambit综合软件，约束定义了芯片的外界。

◆ 对于时序电路，你必须定义一个时钟。

◆ 当输入到来和需要输出时定义。

◆ 定义输入和输出的强度和负载。

2.3.7 设置当前模块和顶层时序模块

◆ 使用这个命令，相对于当前模块约束被置于设计对象上：

set_current_module mod_name

设置当前模块的图形界面如图2.5所示：

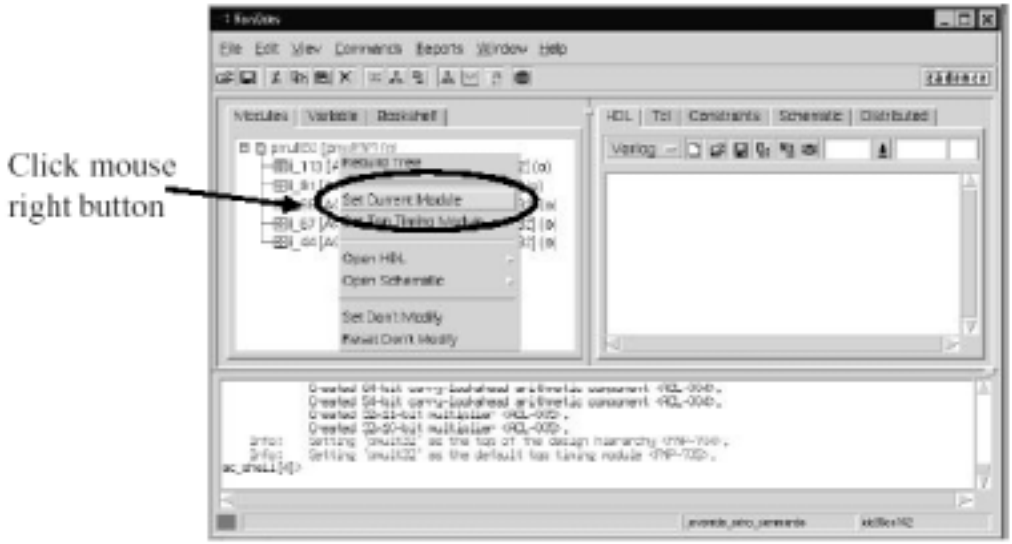


图2.5 设置当前模块

◆ 顶层时序模块用来识别顶层用作时序的模块。所有时序约束通过这个命令在其模型里申请：

set_top_timing_module mod_name

2.3.8 定义时钟

两个步骤

◆ 首先要定义一个理想的时钟

set_clock IDEAL_CLOCK –period 20.0 –wave {0 10}

◆ 然后你必须使一个时钟端口和你的理想时钟联合起来

set_clock_arrival_time –clock IDEAL_CLOCK \–rise 0 –fall 10 clk

2.3.9 设置数据到达和需要的时间

◆ 使用这个命令来设置数据到达时间：

set_data_arrival_time 0.0 –clock \ideal_clock [find –port –noclock –input *]

设置数据到达时间的图形界面如图2.6所示：

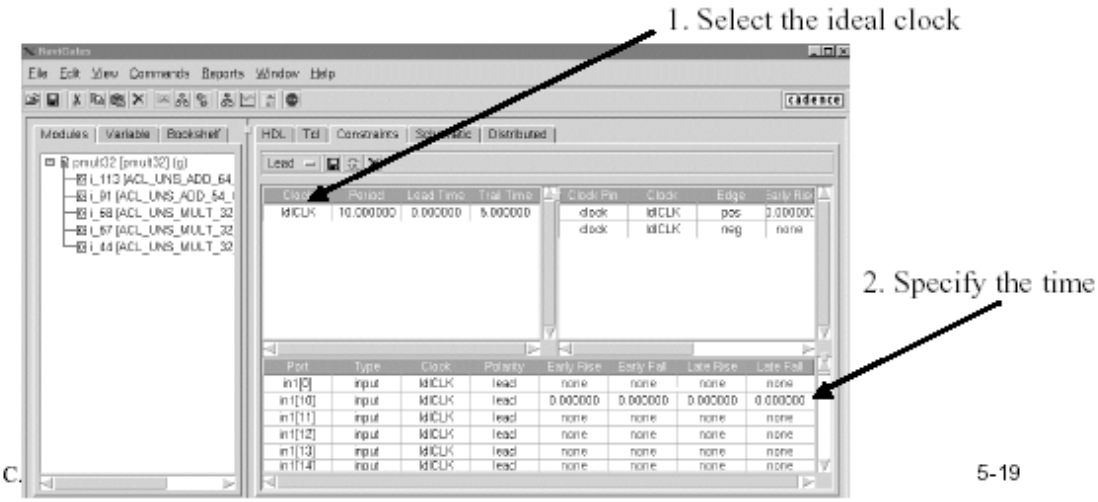


图2.6 设置数据到达时间

◆ 设置需要有效数据在输出端口稳定的时序信息：

set_data_required_time value –clock \ideal_clock[find –port –output *]

2.3.10 输出负载和输入强度

◆ 设置输入驱动电阻

set_drive_resistance value \ [find –port –input *]

◆ 设置输出负载

set_port_capacitance value portlist

2.3.11 检查约束和网表

◆ 检查设计上完全的时序约束：check_timing

◆ 检查设计的网表：check_netlist

2.3.12 进行优化

◆ 使用这个命令开始优化：do_optimize

优化后的门级电路如图2.7所示

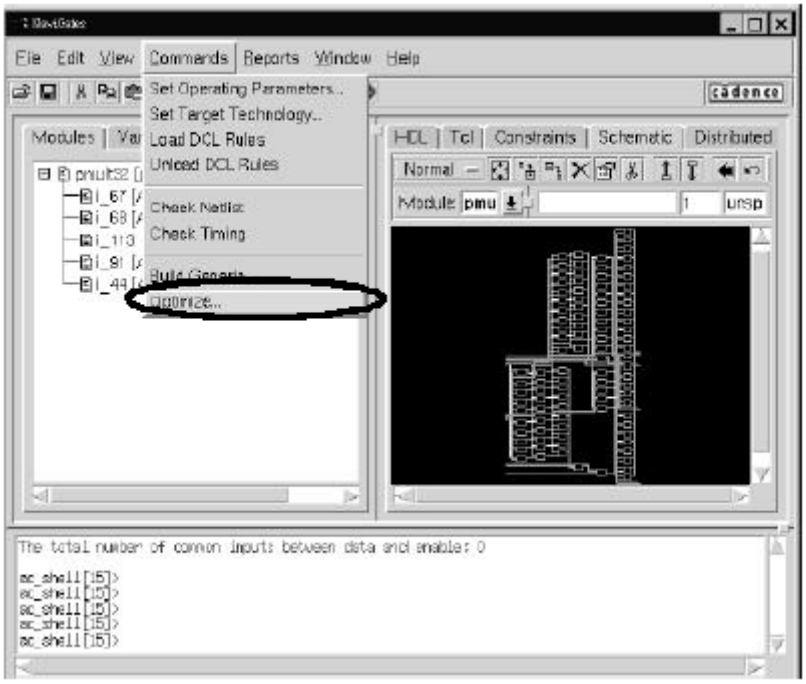


图2.7 优化后的门级电路

2.3.13 产生报告

◆ 两个最普通的报告是关于时序和面积，使用这些命令：

report_area

report_timing

2.3.14 写输出结果

◆ 使用这个命令写门级的文件

write_verilog –hier output_file

（注意：使用 –hier 切换看设计中的每个模块的单元级）

◆ 写数据库格式

write_adb output_file

2.3.15 回顾整个设计流程

1．读入工艺库 read_tlf
2. 设置目标库 set_global target_technology <lib_name>
3. 读入HDL代码 read_vhdl或者read_verilog
4. 产生普通网表 do_build_generic
5. 申请属性 source <constraint> .tcl
6. 映像并优化设计 do_optimize
7. 输出报告 report_timing > file.rpt
8. 写出HDL网表 write_verilog

2.4 设计实例

用一个实例来具体说明使用Ambit进行综合的基本流程

Lab Basic Flow of Ambit Synthesis
Objective：Synthesize your design using Ambit Synthesis.

Design Files：Three modules are used for this design:
mux.v Selects the signal source
ring_plus.v Contains the math operations
bell_box.v Top level model

Running the Design
1. Change directories by entering:
cd lab1_2

2. Enter ambit GUI environment
ac_shell –gui &

3. Enter the key steps in Navigates command window to synthesize the design
(1) Reading in the technology file
read_tlf csmc06core.tlf
read_tlf csmc06pad.tlf
(2) Reading in the Verilog models
read_verilog {bell_box.v mux.v ring_plus.v}
(3)Building the technology independent netlist
do_build_generic
(Double click the bell_box in Module Brower to view the schematic)
(4)Setting the constraints
set top "bell_box”
set_top_timing_module $top

```
set_current_module $top
set_clock IDEAL_CLOCK -period 20.0 -wave { 0 10.0 }
set_clock_arrival_time -clock IDEAL_CLOCK -rise \0.00 –fall 10.00 clock
proc all_inputs {} {find -port -input -noclocks *}
proc all_outputs {} { find -port -output * }
set_data_arrival_time 3.5 -clock IDEAL_CLOCK \[all_inputs]
set_drive_resistance 0 [all_inputs]
set_data_required_time 14.00 -clock IDEAL_CLOCK \[all_outputs]
```

(5) Optimizing the design

```
do_optimize
```

(6) Generating reports

```
report_timing > timing.rpt
report_area -hier -cells > area.rpt
report_design_rule_violations > drc_violations.rpt
report_fanin out > fanin.rpt
report_fanout mode > fanout.rpt
report_hierarchy > hierarchy.rpt
```

(7) Writing out the gate-level models

```
write_verilog -hier bell_box_h.vg
```

Note: you can collect the commands you set ((1) to (7)) to a script file, e.g. (script.tcl). Then use “source script.tcl” in command window to progress the synthesis.

4. Make sure the module tab is selected in the NaviGates browser window (center-left of the main window). Each module is labeled with b, g, m, o, or x. These labels indicate the state of the modules listed.

Module state Definition

b Black box (no subcomponents)
g Contains generic view
m Contains mapped view
o Contains an optimized view
x Module is marked as “don’t modify

5. Right click bell box in the module tab.

a. Select Open Schematic—Main Window.
b. Left click on the module u1.
c. on smart icons above schematic window, click the icon “down hierarchy”, The schematic of u1 is displayed.
d. Select “Up hierarchy”
e. In the schematic window, right click. A popup appears. Select worst path from the popup.
f. To clear the highlighted path, select the smart icon above the schematic window “Click Highlighting”

6. Reviewing the Reports

Review the reports created.

第三章 自动布局布线与 Silicon Ensemble 的使用

3.1 自动布局布线概述

3.1.1 布图规划

规划是将电路放置在一枚专用集成电路芯片上的第一部。其输入文件是一个层次式的网表文件，来自语前端设计或系统分片的输出。层次式网表的内容包括:功能块之间的互连，功能块之中逻辑单元的端点。布图规划设计是将功能块安排在 ASIC 芯片上，是 ASIC 的逻辑表征对应于物理表征。

在布图规划设计中所需考虑的因素如下：

（1） 安排固定功能块的位置，重新调整可变功能块的形状以减小芯片面积；

（2） 规划功能块之间的互连空间；

（3） 决定输入输出 PAD 和电源 PAD 的位置，以及电源 PAD 的数目；

（4） 决定电源和时钟在芯片上的分布；

（5） 减小功能块之间的互连线长度和信号延迟。

其中，布图规划设计中最重要的指标是减少芯片面积和减少延迟时间。

基于单元设计的 ASIC 芯片中的功能块有两种，即可变功能块和固定功能块。利用布图规划设计工具可以对可变功能块进行重新划分，并改变功能块的形状以达到最佳的布图效果。

3.1.2 布局设计

在布图规划确定了固定功能块和可变功能块在芯片上的位置后，布局设计确定所有标准单元在可变功能块中的位置，布局设计的主要目的是便于或优化随后的几何布线设计，同时减少关键节点的互连延迟和芯片面积。

布局设计的输入数据是布图规划设计的输出数据，布局设计的结果将作为随后的布线设计的输入。通常布图规划设计和布局设计的 CAD 软件总是紧密连接在一起的，但布局设计更适合于进行自动设计。在布图设计完成后，我们可以得到一套完整的，包括功能块之间和块内的互连线寄生电容，使我们能够更精确的预计每一逻辑单元的实际负载，这些数据将反注回前端设计。

为便于布局算法估计布局的质量，必须确定布图设计中特殊的可测性的参数。既然布局是基于全局的布线而不是个别的布线路径的优化，因此布图设计的参数优化指标为：

（1） 使总的预测互连长度最小；

（2） 满足关键节点的时序要求；

（3） 减少互连密度。

显然，为满足上述参数优化，布局设计在三者之间作出折衷。

3.1.3 布线设计

布线设计是 ASIC 后端设计过程中的最后一个环节，在完成芯片的布图规划和标准单元的布局设计之后，可以通过对 ASIC 芯片的布线完成所有节点的连接，布线设计过程相当复杂，一般分成两个步骤，首先是全局布线设计，其目的是产生一个布线规划，为每一段互连线段找到对应的布线信道。然后，详细布线设计将完成所有节点连接的几何图形。

3.1.4 自动布局布线的流程

自动布局布线的流程如图 3.1 所示

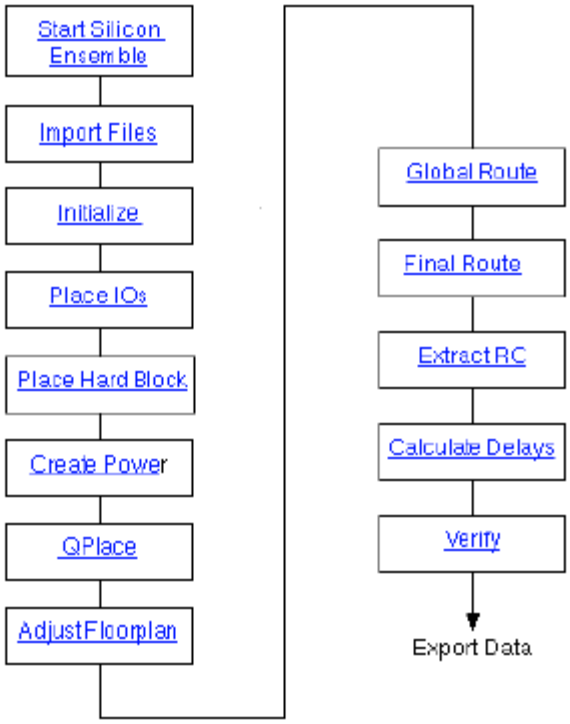


图 3.1 自动布局布线的流程

3.2 工具 Silicon Ensemble 的介绍

3.2.1 Silicon Ensemble的介绍

在这个流程中我们floorplan 和布局布线采用的工具是Silicon Ensemble，简称SE 。Silicon Ensemble 是一个自动版图生成系统。它执行基于标准单元集成电路和子电路设计的平面布置(floorplan),单元放置和互连线的连接。用户既可以使用ANSI(命令行界面)也可以使用窗口界面。Silicon Ensemble 包含几个功能块：平面布置器(floorplanners)，布局器(placers)，布线器(routers)，系统支持工具(system support tools)。

floorplanners: 为器件放置预备行空间。

placer: 包括基于连通性把单元分组，自动放置单元，用Qplace 放置单元，放置单元微调，布局优化等命令。

router: 使用全局(global)，最后(final),能量(power)，时钟，和特定布线器进行布线。当然你也可使用WarpRoute 选项进行快速全局和最后的布线。

system support tools: 让你进行读写数据，验证数据，和手工放编辑单元的放置及连线。

Silicon Ensemble支持多层金属布线(Mutilayer metal routing)，混合库的支持(Mixed library support)， Correct-by-construction layout,自动参数调节(APT),技术更改选项(ECO)。

3.2.2 Silicon Ensemble的用户界面

工具Silicon Ensemble的用户界面如图3.2所示：

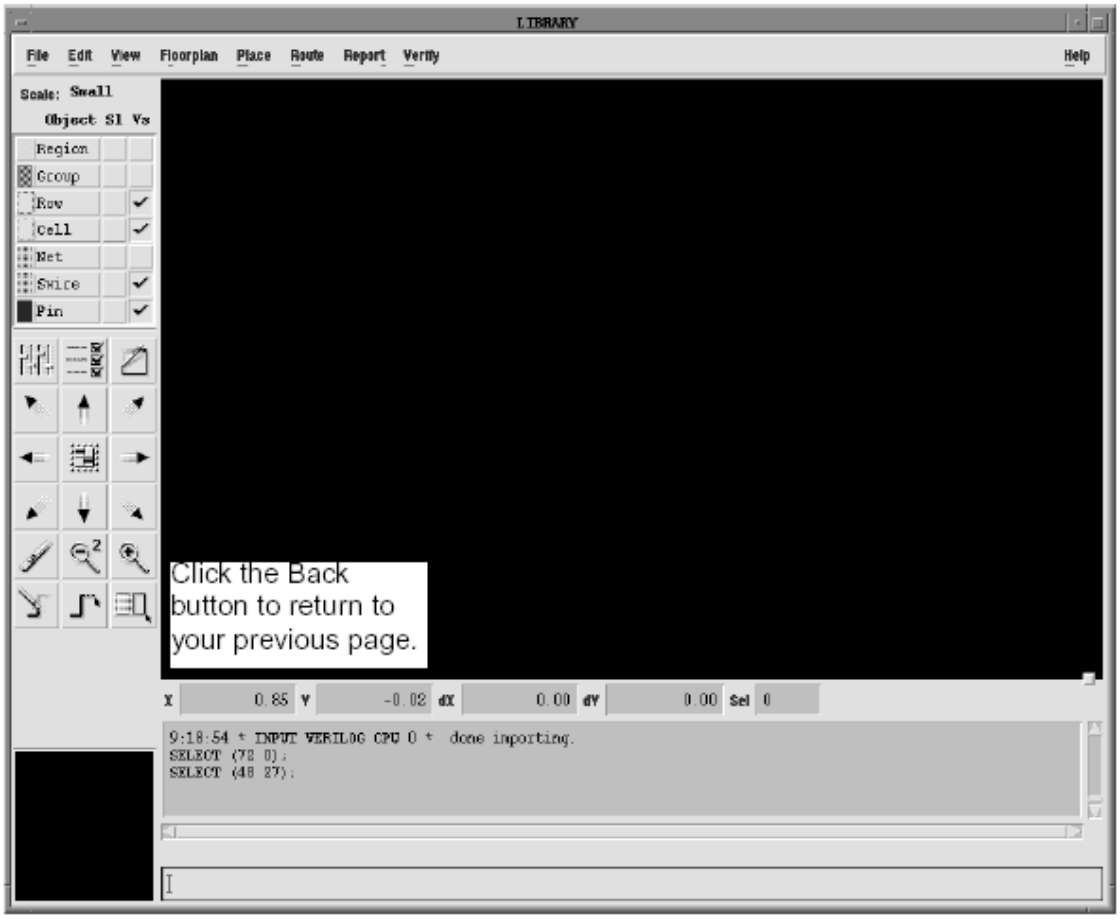


图3.2 Silicon Ensemble的用户界面

3.3 Silicon Ensemble 在自动布局布线中的应用

3.3.1 SE前准备

开始启动SE 工具进行自动布局布线之前，我们需要准备好设计所需的工艺库，设计数据，setup 文件。

1. 工艺库

LEF 文件(Library Exchange Format): 描述了工艺技术和宏单元的ASCII文件

TLF 文件(Timing Library Format)和CTLF 文件(Compiled Timing Library Format)：CTLF是TLF的编译化版本。TLF 包含了库的时序信息。

GCF 文件(General Constraints Format)：包含了压强，电压，温度等要求并指向CTLF时序库。

Verilog 文件：如果你想进行布局优化和生成clock tree,就需要此Verilog。除非你的Verilog网表含有了显式连接或没有模块是有bus pins 的。

2. 设计数据

DEF(Design Exchange Format) 网表:设计数据的ASCII描述。它不仅包含了设计的网表，还包含了设计的物理约束。对于DEF 网表和Verilog 网表，你可以只要一个。当然你也可以在incremental DEF 网表中增加你的附加信息。产生DEF 文件，你可以是自己手工写，也可以是用Preview 之类工具产生。

Verilog 网表：可以在读完此文件以后在读入incremental DEF 文件，来导入附加的设计数据。

GCF(General Constraints Format)文件

SDF 约束文件

3. setup 文件

se.ini: 设置了环境变量，也可以作为一个自动执行的脚本文件。它在SE 工具启动时从工作目录或逻辑目录中读入此文件。

se.env: 设置了系统运行的环境变量。如果你想设置控制系统运行的变量或设置数个用户的工作环境，则需把此文件放在当前工作目录下。软件在启动的时候将在当前工作目录下搜索此文件。

se.fin: 软件在关掉之前读入此文件。

dlc.init: 如果你没有使用GCF文件，为了初始化the Central Delay Calculator(CDC),需要这个文件。

3.3.2 时间驱动设计：

1. 设计流程

时间驱动设计流程如图3.2所示：

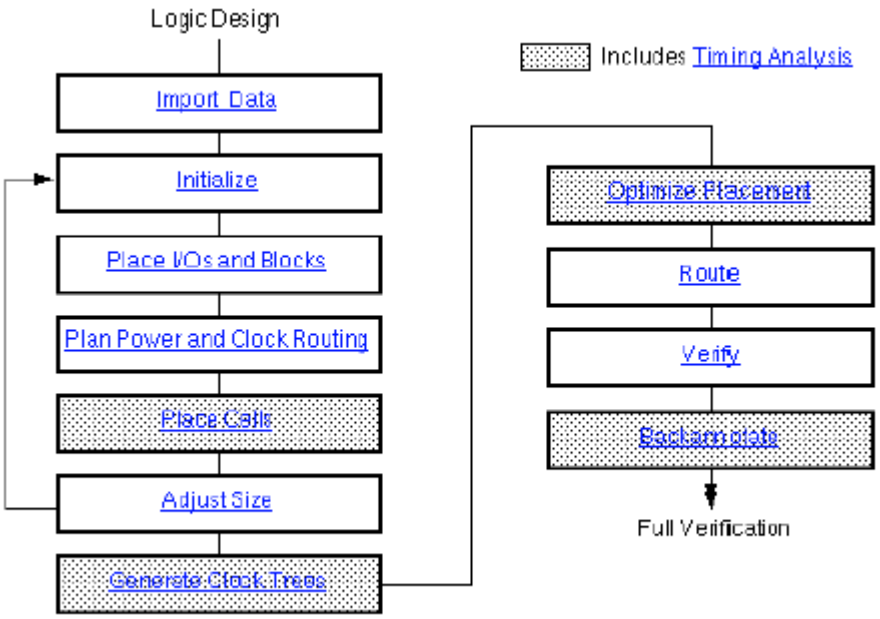


图 3.2 时间驱动设计流程

2. 启动工具

启动SE 工具：

se | sedsm | seultra [-b] [-j=journal] [-gd=drive] [-e=environmentalFile]
[-m=memorySize] [“cmds”] [&] [-version]
-b: 指定运行批处理模式。
-gd=ANSI: 调用非图形界面。提示符为SE>
-gd-x: 调用图形界面。缺省为图形界面。
执行ANSI 的批处理：
se | sedsm =b -gd=ansi “EXECUTE script.com; “ &

3. 导入数据文件

（1）导入LEF 文件： LEF 文件包含了工艺和库的信息。你可以重复读入多个LEF文件。如果你有GDSII 库信息，可以使用AutoAbgen来产生你的库。图形窗口界面操作为：File-Import-LEF；命令行界面操作为：INPUT LEF。后面的命令操作不再具体指出哪个是图形界面操作，哪个是命令行界面操作，凡是有“ - ”就是图形界面操作，否则就是命令行界面操作。

（2） 导入时序库信息：你即可以导入GCF文件也可以导入CTLF 文件。GCF文件里含有指向CTLF 的语句。如果你导入的是CTLF 文件，则你还需dlc.init 文件。命令为：File-Import-Timing Library 或INPUT CTLF 。

（3） 读入Verilog 库文件(可选)：如果你想进行布局优化及生成Clock tree,你就需要读入此文件，否则不必。命令为；File-Import-Verilog或INPUT VERILOG。

（4）读入你的设计网表：读入你的设计网表，你可以有两种方式：

i) Verilog 网表： File-Import-Verilog 或INPUT VERILOG.

ii) DEF 文件:File-Import-DEF 或INPUT DEF。

（5） 读入DEF 文件(可选)；如果你有增加的信息，如有关芯片边缘的单元等信息，你就需要读入含此信息的DEF 文件。这文件一般手写。命令为： File-Import-DEF 或INPUT DEF 。

（6） 读入边界条件(可选)：边界条件是指外部管脚的驱动和负载。对于边界条件你必须使用GCF 文件。边界条件在所有的时序命令中都会被使用。如果你想产生clodk tree ，你必须有此边界约束。你也可以使用同样的GCF 文件读入下面一步中的系统约束。

（7） 读入GCF或SDF格式的系统约束文件：此SDF文件可以是DC产生的约束文件。

（8） 产生Verilog 文件(可选)：此Verilog 文件包含了所有的设计网表信息，包括边缘单元等。它可以用来和逻辑设计比较。

4. 初始的floorplan

为你的物理设计进行初始化。它产生一个只有行或列的core,放置I/O 的行或列则放在core的四周，同时产生GCell 和RGrid 轨迹。它不放置任何单元，只是初步定义了芯片的大小，行的利用率。

命令为： Floorplan-Initialize Floorplan或INITIALIZE FLOORPLAN 。

如果必要你可以手工对行，单元等用EDIT 或MOVE 等等进行编辑。

5. 放置I/O 单元和blocks

一般来说在放置blocks 之前先放好I/O 单元，如果你有放置I/O pads 的约束文件。否则你在放置单元的时候一起放置你的I/O pins 。放置I/Opads 的命令为： Place-I/Os或PLACE IO.

手工(Edit-Move 或MOVE)或自动(Place-Blocks 或QPLACE BLOCKS 放置你的关键 blocks 。

对blocks周围的行进行修剪。Floorplan -Update Core Rows或CUT ROW.

6. 规划电源布线：

Route-Plan Power 打开工具栏。如果有必要你就先编辑电源路径(-Delete Pwr Path或-Restore Pwr Path).然后加入power rings(-Add Rings 或CONSTRUCT RING)和power stripes(-Add Stripes 或ADD STRIPE)。如果有必要你在编辑电源线(RingWires-Add | Change | Delete).

7. 放置单元(cell)

如果必要，你应该先产生放置cell,然后使用时序选项去放置你的cell 。同时为了下面的时序分析，你应该选中产生RSPF 文件的选项。这RSPF 文件是基于全局布线预算的。如果你没有放置I/O pins,则需在放置cell 时选中pin placement 选项。

检查拥挤映像以估计可布通性。如果你的设计没有布通，你必须重新floorplan 。你用Floorplan-Compact Floorplan(Vsize)或VSIZE调整你的设计。

8. 时序分析

SE 使用静态时序分析器和Central Delay Calculator 计算精确的时序信息。用Report-Timing Analysia 或REPORT TIMING 检查你的设计是否满足时序要求。用Report-Timing Path或REPORT TIMING PATH 检查各个路径。

如果你想用第三方的珍珠延时计算器或时序分析工具来验证你的设计是否满足时序要求，你需要输出以下信息并在相应的软件中读入这些信息：

用Report-RC 或REPORT RC 输出寄生信息；

用Report-Delay 或DELAY 输出延时信息，写成SDF文件。

9. 调整尺寸

到了这一步，你可以估计你的芯片大小并在可布通的条件下调整你的设计尺寸。调整设计尺寸大小的命令为Floorplan-Compact Floorplan(Vsize)或VSIZE。

如果你调整的程度每一维都超过了10%，则你需要重新你的floorplan。

10. 生成clock trees(可选)

如果你使用命令行界面，你需要一个CTGen 格式的约束文件。用Place-Clock Tree Generate 或CTGen 设置skew budget 和插入延时限制，然后运行。如果你的clock tree不符合你的skew 限制,你需反复执行这些步骤。接下去再用File-Import-DEF或INPUT DEF ECO 读回clock tree 。

产生报告查看clock tree是不是满足你的要求。用Report-Clock Skew或REPORT CLOCK SKEW 命令检查基于全局布线的clock skew 分析。用Report-Timing Analyzsis 或REPORT TIMING ANALYSIS 命令分析设计的保持建立时间,负载和传输电压。如果有违约(violations)，你可以用布局优化器进行纠正。此时你可以用后注释更新你的逻辑设计。

11. 布局优化(可选)

布局优化重新调整了门的尺寸，插入了缓冲器以纠正时序和电气的violations。进行此步骤的前提是你有时序库文件和GCF 系统约束。布局优化你可以使用PBOpt工具(命令为Place-Placement Optimization-PBOpt 或PBS)，也可以使用QPlace 优化器(Place- Placement Optimization-QPlace Optimization)进行优化。

如果你有很多的缓冲器，则你还需用Floorplan-Compact Floorplan(Vsize)或VSIZE命令进行调整。这时你也可以用后注释来更新你的逻辑设计。

12. 布线

使用Route-Connect Ring 或CONNECT RING命令对power nets 进行布线。

使用Route-Clock Route 或CLOCK ROUTE 命令对clock trees进行布线。

对剩下的nets 进行布线： Route-Warp Route 或WROUTE,或者是按传统的方法：先Route-Globals Route 或GROUTE 再Route-Final Route 或FROUTE 。用Search 和Repair模式可以自动确定一些violations 。

13. 验证

检查时序，方法同上。

用Verify-Geometry 或VERIFY GEOMETRY 命令检查有没有几何或物理的版图violations。

用Verify-Connectivity或VERIFY CONNECTIVITY命令检查有没有连接的violations。

用Verify-Antennas或VERIFY ANTENNA 命令检查工艺antennas 。

用search和repair模式FROUTE，或重新对wire和net进行布线，以确定violations，然后用Edit-Wire 手工纠正这些violations 。

14. 后注释

用Report-RC 或REPORT RC 命令提取RSPF 格式的寄生参数。

用Report-Delay 或REPORT DELAYS 命令写SDF延时文件。

产生新的Verilog 网表文件： File-Export-Verilog 或OUTPUT VERILOG 。

3.3.3 数据路径设计(datapath design)的自动化布局布线

datapath design 的布局布线和时序驱动设计相似。只有微小的区别。就是datapath设计需要用File-Import-Verilog 或INPUT VERILOG 读入.vip 文件来作为Verilog 网表。而在完成floorplan 之后，进行其它操作之前需放置datapath 结构。放置datapath 结构的命令为：

Floorplan-Datapath Toolbox-Initial Functions;

Floorplan-Datapath Toolbox-Add Regions;

Floorplan-Datapath Toolboxl-Place Regions 。

3.3.4 ECO 处理

1. 用File-Import-DEF ECO(INPUT DEF ECO)或File-Import-Verilog(INPUT VERILOG ECO)读进改变了的网表。
2. 根据设计状态决定你是否应该重新布局。
3. 对有所改变的nets 或面积重新布线。

注意： GUI 中小数点后的两位算的，如： GUI中10.00 ，则命令行中将是1000。

由DA 得到的Verilog-HDL 需加入PAD的说明。

附加的DEF和IO constraint file (.ioc)和generate clock tree 的命令文件**.ctgen.cmd需自己写.

3.4 使用SE的一些说明

3.4.1 使用block LEF进行自动布局布线

层次化的布局布线

子模块布局布线完成

导出block LEF文件

上一层布局布线时，导入库文件后即导入前面的block LEF文件

重复上面步骤

3.4.2 Wrap Route的重复进行

当第一次布局布线完，如果verify有错，可以返回到Wrap Route这一步。

选择Search and Repair项，再进行一次布线。

布完后，再verify, 若无错，结束，若有错，重复上一步。

3.4.3 版图报告report

◆ report summary

使用器件数，row利用率，使用连线数，使用pin数目等

◆ report RC

给出每一个net上的估计负载

◆ report delay

给出每一个net上的估计延迟

3.5 设计实例

3.5.1 布局布线前准备

1. 修改DC 生成的Verilog 文件，为其加上core 的pad 及电源与地的pad 的说明。当然也可写成DEF 格式。

2. 编辑IO 的约束文件(.ioc)文件

3. 如果你是用脚本文件做批处理，则还需写好生成clock tree 所需的约束文件(.constraint)和命令文件(.ctgen.cmd)

3.5.2 启动SE 进行布局布线

转到工作目录

%cd work

把启动工具写成一个命令文件InitWorkDir.csh 放在非工作目录scripts/：

```
*******************************************************************************
#!/bin/csh -f
if(`basename $cwd` == "work") then
# Remove all previous files.
\rm -r dbs *.cfg *.dtp *.gds *.info *.ini *.jnl *.rpt *.summary *.wdb
\rm -r *
# Prepare the directory.
mkdir dbs
```

```
cp ~chwtang/CSMC/csmchdlib/se/se.ini ./se.ini
# Start se in the background.
sedsm -m=500 &
else
echo "Not in work directory."
endif
```

****

执行启动工具的命令文件：

% ../scripts/InitWorkDir.csh

启动了SE 后读入数据文件：

File-Import-LEF ：读入csmc06.lef ；

File-Import-Timing Library ：读入csmc06.gcf ；

File-Import-Verilog：读入设计文件ddfs.v，ddfstop.v和库的verilog文件csmc06.v；

File-Import-SDF：读入SDF 文件ddfs_constraints.sdf ；

File-Import-DEF ：读入电源和地的pads DEF 文件；

到这里为止， SE 窗口都看不到什么东西。

下面各个参数和变量的设置，这里不再详述。

Initialize Floorplan ；

Floorplan-Initialize Floorplan ：设置各种参数

SE 窗口中出现rows 。

放置Iopads:

Place-Ios：读入IO pads 的约束文件ddfstop.ioc；

SE 窗口中显示出IO pads 的放置。

power planning；

Route-Plan Power-Add Rings；

Route-Plan Power-Add Stripes；

放置标准单元：

Place-Cells：设置好时序变量。生成clock tree:

Place-Clock Tree Generate(CTGEN)：设置clock tree 的约束文件，产生clock tree 。

读回clock tree 的def 文件：

File-Import-DEF ECO；

place filler core cells；

Place-Filler Cells-Add Cells ；

布线：

Route-Connect Ring；

Route-Wroute；

输出各种文件：

File-Export-GDS II：输出gdsII 网表；

File-Verilog：输出Verilog 文件用于版图后仿真；

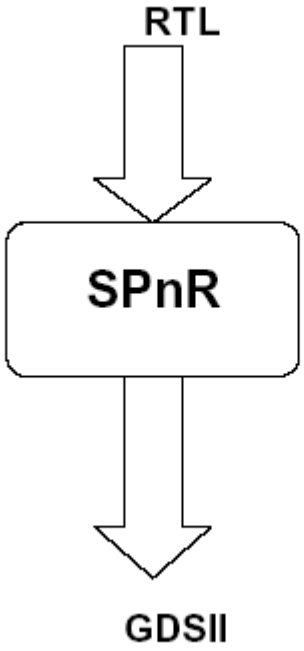
Report-Delay：输出SDF 文件，用于版图后仿真。

第四章　PKS(Physical knowledge synthesis)的使用

4.1 物理综合流程的介绍

4.1.1 从 RTL 到 GDSII

本文即将介绍 Cadence 公司的新的 SPnR(Synthesis Place and Route)综合布局布线流程，它能够让你进行完全从 RTL 到 GDSII 的复杂深亚微米数字集成电路设计。



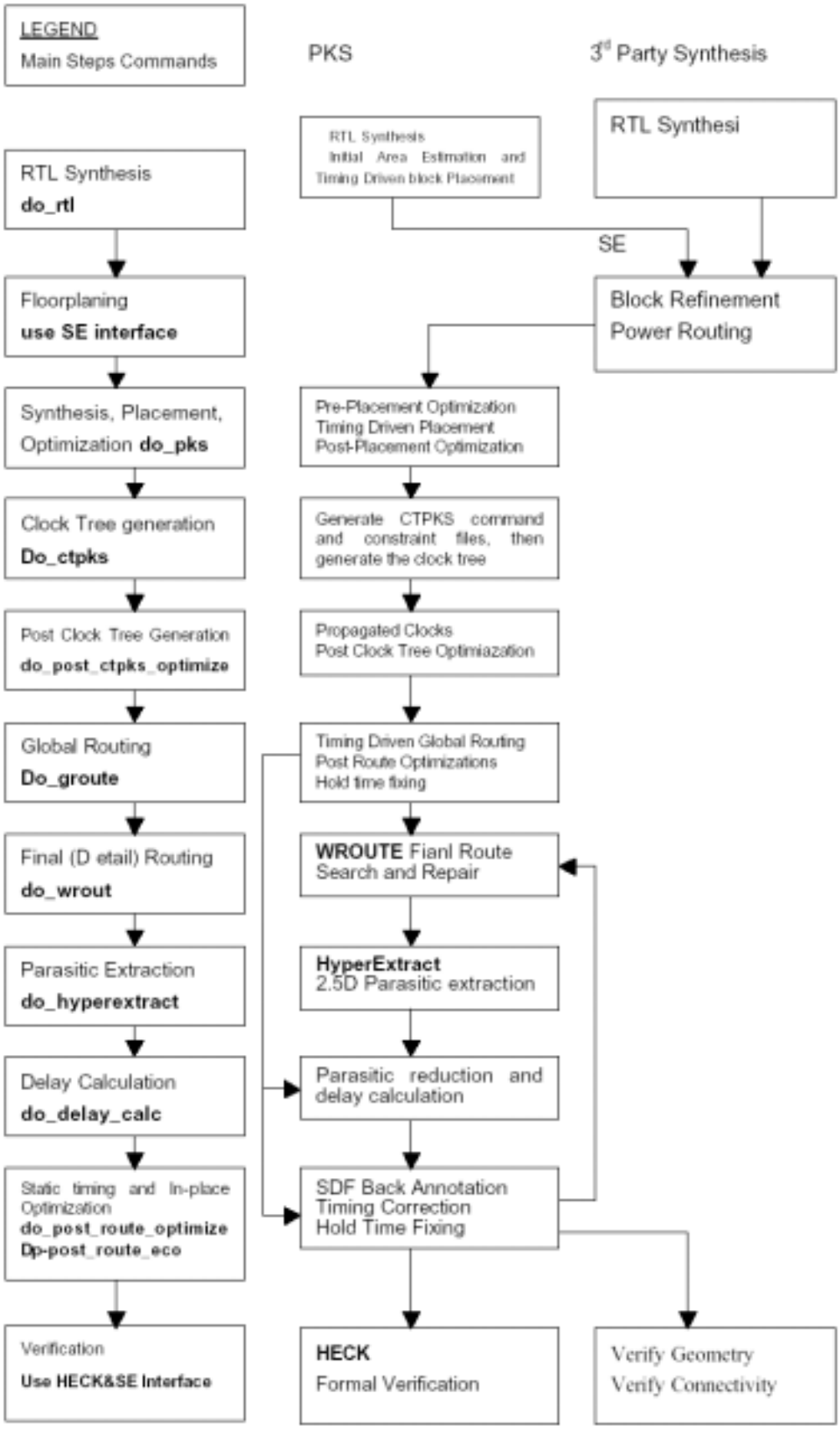
这种新的 SPnR 流程已经发展到主要应用于深亚微米领域的时间中止和设计可靠性。从事这些工作要求逻辑设计和物理实现工艺上的更高程度的整合。Cadence 公司的 SPnR 流程把 RTL 寄存器和晶体管级综合和最终布局布线紧密地结合了起来。并且照已有的经验，它已经被证明这个新的流程在整个设计流程的各个阶段导致了更小的设计，更大的时间相关性，更短的设计时间。

一个设计有很多种方法可以从 RTL 转换到 GDSII，所以我们已经描述一个典型的流程—可以按照需要对这个流程进行修改。对应于这个流程的每一步，我们详细的说明了需要的输入和结果输出，命令文件，建立要求和推荐的联系。我们还有很多可提供的使 SPnR 流程每个步骤自动化的效用的描述，怎样和哪里他们被使用以及怎么找到他们的信息。同时提供一个全面的相关于流程每个步骤的文档列表（包括应用程序注意）来使你可以更快的按照要求找到你想要的更详细的信息。

为了简单的目的，我们可以假设在 RTL 级的逻辑设计是使用 Verilog 来表示，但是如果需要的话，它也能用 VHDL 来表示。同样的，我们可以假设任何的门级网表可以用 verilog 来表示，但是他们也可以用 VHDL 或者 EDIF 表示。

4.1.2 整个流程和附属命令

下面这个全面的流程图显示了在 SPnr 流程—PKS(Physically Knowledgeable Synthesis)和 SE(Silicon Ensemble)和它们间的高层交互内的两个主要工艺。


LEGEND
Main Steps Commands
PKS
3rd Party Synthesis
RTL Synthesis
Initial Area Estimation and Timing Driven block Placement
RTL Synthesi
RTL Synthesis
do_rtl
SE
Floorplaning
use SE interface
Block Refinement
Power Routing
Synthesis, Placement, Optimization do_pks
Pre-Placement Optimization
Timing Driven Placement
Post-Placement Optimization
Clock Tree generation
Do_ctpks
Generate CTPKS command and constraint files, then generate the clock tree
Post Clock Tree Generation
do_post_ctpks_optimize
Propagated Clocks
Post Clock Tree Optimiazation
Global Routing
Do_groute
Timing Driven Global Routing
Post Route Optimizations
Hold time fixing
Final (D etail) Routing
do_wrout
WROUTE Fianl Route Search and Repair
Parasitic Extraction
do_hyperextract
HyperExtract
2.5D Parasitic extraction
Delay Calculation
do_delay_calc
Parasitic reduction and delay calculation
Static timing and In-place Optimization
do_post_route_optimize
Dp-post_route_eco
SDF Back Annotation
Timing Correction
Hold Time Fixing
Verification
Use HECK&SE Interface
HECK
Formal Verification
Verify Geometry
Verify Connectivity

4.2 PKS 在 SPnR 流程中的应用

4.2.1 启动前准备

在你开始 SPnR 流程之前，你必须有：

1．特定的文件和库。SPnR 流程最初需要的文件和库有下面这些：

（1） 逻辑（RTL）设计数据 (必需)

支持格式：Verilog 和 VHDL

（2） 物理设计数据（可选）

支持格式： DEF 5.1+, PDEF 2.0

（3） GCF 文件（可选）

支持格式： GCF 1.3, 1.4

（4） 时序库 （必需）

支持格式： ALF 3.0, OLA(DCL), TLF 4.3

（5） 时序约束 (必需)

支持格式：TCL

（6） 物理库 （必需）

支持格式： LEF 5.2 或者 LEF 5.3

（7） 层利用目录 （必需）

支持格式： ASCII 文本

2. 运行脚本和封装脚本

设及流程的每一步中能够通过输入单独的指令来进行人工控制。但是这个方法明显易于犯错。所以推荐你进行运行脚本歌封装脚本来定义流程环境并载入适当的库和设计数据。

Run script:	do_rtl	进行 RTL 综合
Run script:	do_pks	进行综合，布局和优化
Run script:	do_ctpks	进行时钟树产生
Run script:	do_post_ctpks_optimize	进行后时钟树优化
Run script:	do_groute	进行全局布线和优化
Encapsulation:	do_wroute	进行最终布线
Encapsulation:	do_hyperextract	进行寄生参数提取
Run script:	do_delay_calc	进行延时计算
Run script:	do_post_route_optimize	进行后布线优化
Encapsulation:	d0_post_route_eco	进行后 ECO

3. 命令文件：

setup.tcl, library,tcl 和 design.tcl

4. PKS 参数的普通设置

一些典型的例子包括：

set_global placement_initialize_autopass true

set_min_wire_length 10

set_Steiner_mode 1

还要知道在 4.0 版本里，一些变化被设定为默认设置。这些新的默认设置如下（使用 report_globals 命令查看全局变量设置）：

auto_slew_prop_selection	true
path_style_constraints	true
clock_gating_to_be_checked	true
extra_space_for_opt	0

4.2.2 RTL 综合

PKS 也能够用于产生一个初始的布图（存在的物理数据可能可选择的被加进了 DEF 文件中）。在 SPnR 流程的这个步骤中，芯片的面积将会被决定，象功能块，垫等混合组件的位置将会被确定，并且那些宏单元也将被安置。

RTL 综合步骤的主要操作是：

- 输入 RTL 设计。
- 在 PKS 的参数里输入布图的信息。
- 执行综合和布局。
- 输出一个门级的 Verilog 网表以及一个包含初始布图信息的 DEF 文件。

RTL 综合的输入和输出

这一步骤的输入如下：

- <library>.tlf TLF 时序库数据文件
- <library>.lef LEF 物理库数据文件
- <library>.lut 一个 LUT 层利用目录
- demo.v 原始的 RTl 源文件
- demo.def 原始的物理数据文件
- constraints.tcl 原始时序约束 TCL 文件
- floor.tcf 布图 TCL 文件

这一步骤的输出如下：

- demo_rtl.adb Ambit 数据库，包含设计，布图和约束信息
- demo_rtl.v 整个设计层次化的门/宏级网表
- demo_rtl.def 包含了初始化布图和电源设计信息的物理数据
- timing_rtl.txt 时序报告文件

RTL 综合使用命令： do_rtl

4.2.3 布图

这是 SPnR 流程里宏功能块和垫被放置到最终位置的步骤。电源格在这一步里也会被加进去。注意：你可以使用象 PKS 或者 DP 来进行布图工作，但是在这个 SPnR 流程是基于使用 SE 来进行具体布图的。

在布图中主要执行的操作是：

- 载入设计 读入 LEF 和 DEF 文件。如果进行时序操作，需要读入 TLF 文件。
- 安置功能块的位置 使用移动单元功能，就可以设置宏功能块的最终位置。
- 加入单元 加入电源垫和一些布局前的装填单元。
- 电源规划 使用 SE 中的 power planner，在设计里创建主要电源总线的格并在宏周围创建电源环。
- 输出新的 DEF 文件 完成布图以后，就会输出一个新的 DEF 文件，包含了垫，宏功能块，电源布线和加进去的布局前装填单元的最终布局。

布图的输入和输出

这一步骤的输入如下：

◆ <library>.lef　　LEF 物理库数据文件。

◆ demo_rtl.def　　物理数据文件（在 RTL 综合中产生）。

这一步骤的输出如下：

◆ demo_se.def　　新的包含布图和电源设计信息的物理数据

4.2.4 综合，布局，优化

一旦在布图步骤里你已经结束电源规划并确定了宏功能块的布局，你就可以开始进行布局和优化的网表。

在综合，布局和优化里的重要操作是：

◆　输入已经升级的布图和电源规划的 DEF 文件。

◆　设置适当的 PKS 控制。

◆　执行时序驱动布局。

◆　进行优化。

◆　输出新的门级网表文件以及一个新的 DEF 文件用于时钟树产生步骤。

综合，布局，优化的输入和输出

这一步骤的输入如下：

◆ <library>.tlf　　TLF 时序库数据文件

◆ <library>.lef　　LEF 物理库数据文件

◆ <library>.lut　　LUT 层利用率表

一下步，如果原始的 RTL 没有装换并且你有好的时序，你可以使用下面的输入文件：

◆ demo_rtl.adb　　层次化的门/宏级的网表（RTL 综合中产生）。

◆ demo_se.def　　物理数据文件（RTL 综合中产生）。

作为选择，如果原始的 RTL 文件已经被转换，或者你想从基于你的新物理数据的 RTL 综合开始做起，你可以使用下面的输入文件：

◆ demo.v　　原始的 RTL 源文件。

◆ demo_se.def　　物理数据文件（在步图步骤中产生）。

◆ constraints.tcl　　原始时序约束 TCL 文件。

这一步骤的输出如下：

◆ demo_pks.adb　Ambit 数据库，包含设计，布图和约束信息。

◆ demo_pks.v　　新的整个设计的层次化的的门/宏级网表

◆ demo_pks.def　新的物理数据文件，包含布图，布局和电源设计信息

◆ timing_pks.txt　　时序报告文件

综合，布局，综合使用命令：do_pks

注：如果以已经优化过但还没有布局的网表开始 PKS，你不需要进行任何布局前优化，只需要做： do_place　　do_xform_optimize_slack

4.2.5 时钟树产生

时钟树综合（CTPKS）已经被整合进 PKS4.0 版本。它使用和 CTGen 完全一样的核心算法，然而时序，寄生，布局和路径显示是由 PKS 代替了 CTGen 自己的引擎提供。

如果已经存在一个 CTGen 类型约束，可以使用 ctgen2pks 将其转换成 CTPKS　TCL 类型约束。新的程序（do-ctpks）已经提供来产生基于设计约束的约束。

使用 CTPKS 产生时钟树的条件：

- 时序库文件（TLFALF），一个物理库文件或者文件(LEF)和层利用表（LUT）必须先载入。
- 设计必须已经被完全并合法的布局。
- 如果多路复用器用来选择一些时钟中的一个，这个“ set_constrain_for_timing ”必须被设置在多路复用器的管脚上，用来选择哪个时钟将被用于时序综合。
- 所有时钟源必须通过“ set_clock_root ”被详细说明。时钟延迟通过“ set_clock_insertion_delay ”被详细说明。
- 时序约束必须被载入设计。
- 在逻辑库显示但在物理库里丢失的组件必须在约束文件中被设置“ set_cel_property don’t_utilize true {$celllrefs} ”。
- 默认设置，所有的缓冲器和反相器都是可用的。你必须禁止你不需要时钟树工具利用的单元(这一点和 CTGen 的工作方式不一样)。
- 使用“ get_clock_tree_objects –buffer|-inverter　”列出时钟树不可用缓冲器和倒相器。
- 使用“ set_attribute $cellrefs ct_don’t_utilize true ”列出 CTPKS 中不用的单元。
- 使用“ set_attribute $cellrefs ct_don’t_utilize fause ”希望得到在 CTPKS 中被使用的单元。
- 确信 dont_modify 特性没有被设置在时钟网络和时钟源的任何单元上。

时钟树的产生：

- 在运行目录创建一个名为 ctpks.tcl 的 TCL 约束文件。
- 用给定的时钟源产生默认时钟树约束。
  - ◇ set_clock_tree_constraints –pin $clock_pin –min_delay 2.0 –max_dealy 2.5 –max_leaf_transition 1.0
  - ◇ 设置时钟树边界
  - ◇ 把“ do_build_clock_tree –noplace –pin $clock_pin –save_structure $clock_structure ”写入约束文件里。
  - ◇ 写出“ report_clock_tree –pin $clock_pin >clock_tree.rpt ”
  - ◇ 写出“ report_clock_tree_violations –pin $$4clock_pin > clcok_tree_violation.rpt ”。
- 在约束文件产生后，需要的话，你可以对“ set_clock_tree_constrains ”中指定的值进行适当的修改。然后你可以用这个约束文件产生时钟树。使用“ report_clock_tree_violations ”输出报告来看时钟树约束是否符合。不过不符合，可以改变设计的布局，更改时序约束和重新产生 CTPKS 约束文件和新的时钟树，或者你可以释放时钟树约束再重新运行 CTPKS 直到标准符合。
- 如果时钟树被正确的建立，运行“ do_place –eco ”来使新插入的时钟树缓冲器合法化。注意：do_place –eco 只需要运行一次。
- 在执行“ do_build_clock_tree ”命令中，“ set_clock_propagation_mode ”将自动设置成“ 已传播的 ”，并将保持这个设置直到你重新设置。

CTPKS 流程图，如图 4.3 所示：

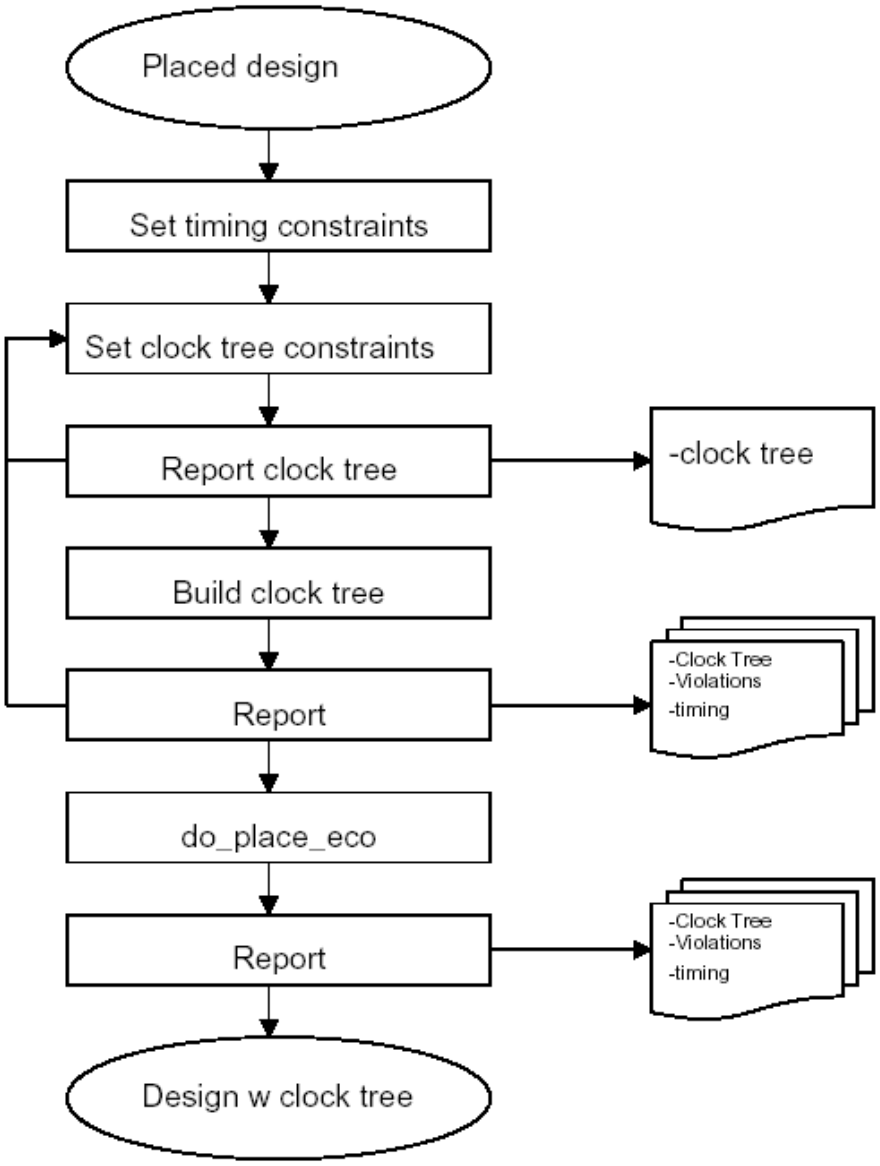


图 4.3 CTPKS 流程图

4.2.6 后时钟树优化

这个步骤需要任何设计的后时钟树优化建立违背规则。这也是保持时间被修正的第一步。在传播时钟模式的优化中，时钟网络的所有成分在布局和时序上都会被修正。

后时钟树优化步骤的主要操作有：

- 进行优化（随着时钟网络修正，优化不能改变时钟树的任何成分，或者任何连接到时钟树的连续的成分）。
- 修正保持时间。
- 输出新的门级 Ambit 数据库以及用于全局布线步骤的 Verilog 网表和新的 DEF 文件。

后时钟树优化的输入和输出：

这个步骤的输入如下：

- <library>.tlf　　TLF 时序库数据文件
- <library>.lef　　LEF 物理库数据文件
- <library>.lut　　LUT 层利用表
- constraints.tcl　　原始的时序约束 TCL 文件

这个步骤的输出如下：

- demo_pks.adb　　包含了设计，布图和约束信息的 Ambit 数据库文件

后时钟树优化使用命令：do_post_ctpks_optimize

4.2.7 全局布线和后全局布线优化

这一步骤需要运行在 PKS 时序环境 WROUTE 的全局布线部分。全局布线优化最初的时序是基 PKS 工具完成的快速布线。由于后面的优化经过全局布线，时序是基于来源于真实布线的 RC 信息。

在全局布线中，布线器根据提供和要求的布线轨迹对 NETS 部分中的定义的规则网络布线。

一旦布线完成，全局布线的 RC 信息会作为每个网络的预报回注到 PKS。预报是估计的 RC 值和实际全局布线 RC 值的差别。任何影响网络的变化都要通过估计计算 RC 值，然后重新申请预报（变化值）。

全局布线步骤的主要操作是：

- 输入 Veilog 网表和由后时钟树优化步骤产生的 DEF 文件或以前保存的.adb 文件。
- 执行现场时序修正。
- 执行全局布线
- 输出一个新的门级 Verilog 网表以及用于后面最终布线步骤的 DEF 文件。

全局布线和后全局布线的输入和输出

这个步骤的输入文件如下：

- <library>.tlf　　TLF 时序库数据文件
- <library>.lef　　LEF 物理库数据文件
- <library>.lut　　LUT 层利用表

然后你可以使用前面步骤产生的 Ambit 数据库文件

- demo_post_ctpks.adb　包含设计，布图和约束的 Ambit 数据库文件（在后时钟树优化步骤中产生）。

作为选择，你也可以使用

- demo_post_ctpks.v　　层次化的门/宏级网表（后时钟树优化步骤中产生）。
- demo_post_ctpks.edf　物理数据文件（后时钟树优化步骤中产生）。
- constraints.rcl　　原始的时序约束文件。

这个步骤里产生的输出文件是：

- demo_groute.adb　　包含了设计，布图和约束信息的 Ambit 数据库文件。
- demo_groute.v　　整个设计新的层次化的门/宏级网表。
- demo_groute.def　　包含布图，电源设计，时钟树和全局布线信息的物理数据文件。
- demo_groute.wbd　　二进制 WROUTE 数据库文件。

保持时间违背修正

在这步里也能进行保持时间违背的修正。如果你的时序库包含的既有最好的也有最差的情况，你可以简单的改变库的模式。如果不是，工具会不得不被重新启动来清除库并且设计的 Verilog 和 DEF 文件不得不被重新载入。

- 如果时序库包含最好和最差的情况：
  - 把操作条件改到最好情况
  - do_xform_fix_hold –incremental –dont_reclaim –critical_ratio 0.0
- 如果要得到最好的情况，一个预备的库必须被读入：
  - write_verilog: 在优化结束后，保存设计的 verilog 文件。
  - write_def: 在优化结束后，保存布局信息。
  - quit and restart: 重新启动 PKS，并载入最好情况的时序库。
  - read_verilog: 读入设计，建立普通连接并写约束。
  - read_def: 读入布局信息进入 PKS 模式。
  - do_xform_fix_hold –incremental –dont_reclaim –critical_ratio 0.0

4.2.8 最终（详细）布线

在这一步骤里布线器使用在 WROUTE 数据库文件里描述的信息布置已经布局和全局布线好的组件。

在详细布线步骤里的主要操作是：

- 用关联的配置文件调用 WROUTE。

有两个级别的布线工艺

- WROUTE 布线器： 这个布线器提供高质量的布线，最多能布六层金属布线。这个 wroute 布线器可以作为 SPnR 流程的一部分使用，可以从 SE GUI 运行，或独立使用。
- ULTRA 布线器： 这个布线器扩展 WROUTE 的性能到 9 层布线。它也能作为 SPnR 流程的一部分使用，可以从 SE GUI 运行，或独立运行。

最终布线由最终布线和查找—修复布线。查找—修复布线中，布线器查找并更正规则违背。

详细布线的配置文件和输入输出文件

这个步骤的输入文件如下：

首先我们需要标准库文件：

- <library>.tlf TLF 时序库数据文件
- <library>.lef LEF 物理库数据文件
- <library>.lut LUT 层利用表

然后，我们需要配置文件：

- wroute.wrconfig 一个 WROUTE 配置文件（由 do_wroute 封装程序自动产生）。

最后但不是最需要的，你将会使用下面的文件：

- demo_groute.wdb 二进制的 WROUTE 数据库文件（由全局布线步骤产生）。

这个步骤典型的输出文件是：

- demo_groute.def 包含布图，电源设计，时钟树和最终布线信息的物理数据。
- demo_wroute.wdb 二进制的 WROUTE 数据库文件。
- demo_wroute.log 日志文件。

最终布线的命令：do_wroute

4.2.9 寄生参数提取

在这个步骤里，将要使用 HyperExtract 工具来计算设计在功能块和芯片级上所有网络间内连的电阻和电容。

寄生参数提取步骤里的主要操作是：

◆ 用相关联规则文件从 PKS 界面调用 HyperExtract。

寄生参数提取的输入输出

这个步骤的输入文件是：

◆ <library>.lef LEF 物理库数据文件

◆ demo_wroute.def 包含布图，电源设计，时钟树和最终布线信息的物理数据文件。（这个文件在最终布线步骤内产生）

◆ hyperextract.rules 这个规则文件是由支持主库的人提供，一般在库目录里。

这个步骤的输出文件是：

◆ demo_hext.dspf 标准的寄生格式（Parasitic format DSPF）文件。

◆ demo_hext.rspf 减少的标准寄生格式（RSPF）文件。

◆ demo_hext.log 一个 HyperExtract 运行日志文件。

寄生参数提取的命令：do_hyperextract

4.2.10 延时计算

延时计算步骤只是在这里举例说明这个流程。它用于在基于 PKS3.0.X 的 SPnR 流程中。在新的基于 PKS4.0.X 的 SPnR 流程中，延迟计算在 PKS 内部运行，没有专门的运行程序或者延迟计算任务来执行。作为替代，寄生减少和延时计算现在在 BulidGates/PKS 中实现。需要的是来自于 WROUTE 或者 HyperExtract 的 DSPF 或者 RSPF 文件。

延时计算的命令：do_delay_calc

do_delay_calc 操作：

◆ 需要一个 SPF 文件（在寄生参数提取步骤中产生）。

◆ 读入 SDF 寄生参数文件。

◆ 报告时序。

4.2.11 静态时序和现场优化

一旦一个布过线设计的标准寄生文件（DSPF 或者 RSPF）已经被创建，你可以线运行静态时序分析和，如果必要的话，执行附加的现场优化。做这个的话，你将读入 SPF 文件来建立网表的预报。这些预告用来保持 PKS 网表预告和网表真实延迟之间的差别的轨迹。

工程变化命令（Engineering Change Order ECO）是读入新的逻辑网表，比较已存在的版图并改变版图不同的地方的流程。在 ECO 布线中，布线器对已经修改的网络进行布线。

整个 ECO 布线中，布线器移动已经被修改的网络的布线并进行重新布线。同样的，在 ECO 布线中，布线器对已经布过线的设计进行最终布线。

ECO 输入文件和网表 DEF 文件有相同的语法。ECO 系统自动决定新网表和内存中数据库的不同。所有这些不同引起 ECO 运行。例如：ECO 删除在数据库中出现但新网表中没有的网络。ECO 也加上在新网表中出现但设计里没有的网络或者单元。因此，新的网表必须是完整的，甚至假如你对相同的数据库做连续的改变。

静态时序和现场优化步骤的主要操作是：

- 载入 SPF 寄生文件。
- 检查时序。
- 执行后布线优化。
- 对那些网表没有被修改的部分进行布线。

静态时序输入和输出

这个步骤的输入文件如下：

- <library>.tlf TLF 时序库数据文件
- <library>.lef LEF 物理库数据文件
- <library>.lut LUT 层利用表
- demo_groute.v 整个设计层次化的门/宏级网表（全局布线步骤中产生）
- demo_groute.def 包含布图，电源设计，时钟树和全局布线信息的物理数据文件。
- demo_hext.rspf 来自于 HyperExtract 寄生文件.

这个步骤的输出文件是：

- timing_ipo.rpt 时序报告。

静态时序的命令： do_post_route_optimize

现场优化的输入和输出

这个步骤的输入文件如下：

- <library>.tlf TLF 时序库数据文件。
- <library>,lef LEF 物理库数据文件。
- <library>.lut LUT 层利用表
- wroute_eco.cmd 自动由封装程序创建的命令文件。
- demo_groute.v 整个设计层次化的门/宏级网表（全局布线步骤中产生）。
- demo_groute.def 包含布图，电源设计，时钟树和全局布线信息的物理数据文件。

这个步骤的产生的输出文件是：

- demo_ipo.adb 包含设计，布图和约束信息的 Ambit 数据库文件
- demo_ipo.v 整个设计新层次化的门/宏级网表。
- demo_ipo.def 包含布图，电源设计，时钟树和全局布线信息的新物理数据文件。

现场优化的命令： do_post_route_eco

4.2.12 验证

在设计完成后，你需要去验证它。首先你可以快速检查一下是不是所有网络都布线了，以及你可以执行预备的 DRC 检查去检查物理布线。以上两种检查都可以在 SE 中执行（具体请见 SE 用户手册）。

在验证中主要的操作是：

- 几何检查
- 连通性检查
- 使用 HECK 来执行正式的验证（具体请见 HECK 用户手册）。

第五章 总结

随着集成电路集成度的不断提高，对集成电路的设计提出了越来越高的要求，设计的芯片必需具有高性能和高保密性，同时设计周期应尽可能短。综合方法是指电路从较高级别的描述自动地转换到较低级别描述的自动设计方法，它已成为电路设计自动化的关键技术。

而“ 物理级综合（Physical Synthesis）” 解决方案则是随着集成电路技术朝着深亚微米发展提出来的新技术。它是指随着半导体制造技术步入深亚微米时代，IC 设计已不能以前后端来划分，而必需将逻辑设计和物理设计结合在一起。

在本文中使用了 Cadence 公司的三个软件 Ambit, Silicon Ensemble, PKS(Physical knowledge synthesis)。Ambit 主要是应用于逻辑综合（logic synthesis）中，Silicon Ensemble 主要是应用于自动布局布线（placement & routing）中。而 PKS 则是随着深亚微米技术的发展，推出的针对物理级综合解决方案的软件。所以它也是我们这里所讨论的物理综合流程研究的重点。PKS 软件涵盖了整个逻辑综合和自动布局布线流程，将逻辑综合和自动布局布线有机的结合了起来，实现了“ 物理级综合（Physical Synthesis）”。逻辑设计和物理设计结合起来后，物理设计出现问题可以很方便的反馈回到逻辑设计，可把逻辑和寄存器传输级设计和综合与版图规划、布局和布线紧密联系起来，形成一个单一的流程，有利于工程师在单一通道里写入 RTL 和执行综合、布局和布线，从而解决深亚微米加工工艺的时序、功率和信号完整性问题。这种逻辑设计和物理设计相结合的设计方法更符合深亚微米集成电路设计发展的要求。这也是未来 EDA 设计发展的趋势。

致谢

在论文写到这里，行将结束的时候，我非常高兴能有这么一个机会在这里表达我对大家的谢意。

首先要感谢我的指导老师--唐长文学长。感谢唐长文学长从论文选题到进行设计直至论文撰写对我的帮助。设计中，唐长文学长努力为我提供各种相关的参考资料。当我遇到疑问和困难的时候，唐学长又以丰富的经验指导我分析解决问题。在此对他致以诚挚的谢意。

还要感谢在大学四年里给过我帮助的所有老师和同学，大家的帮助让我受益匪浅。

最后我还要感谢实验室为我们提供了这么好的软硬件环境，感谢 ASIC 实验室里管理机房的老师和系统管理员。

参考文献

[1] Ambit manual Cadence Design System, Inc.
[2] Documents about Ambit NSC Chip Implementation Center
[3] SE_tutorial Cadence Design System, Inc.
[4] SPnR Flow Guide for PKS 4.0.5 Cadence Design System, Inc.