数字集成电路综合及物理设计阶段的时序收敛方案

谢扬，桑红石

（华中科技大学,自动化学院,多谱信息处理技术国家级重点实验室，湖北 武汉 430074）

摘要：时序收敛是数字集成电路设计中最重要的任务之一。随着集成电路设计进入了深亚微米时代，芯片规模不断增加，设计日趋复杂，时序收敛的难度也随之越来越大。本文基于图像中低层处理 SoC 的综合乃至物理设计，着重讨论在此阶段使时序收敛的方法。本文首先介绍 Design Compiler、IC Compiler 等 EDA 工具的时序分析方式，随后讨论如何利用工具对设计设定合理的约束，最后介绍了各个阶段出现时序违例的解决方式。通过此方法最终使芯片的时序达到收敛。

关键词：集成电路设计；时序；综合；物理设计

中图分类号：TN492

A Method to Achieve Timing Closure During Synthesis and Physical Design Stage of Digital Integrated Circuit

Xie Yang, Sang Hongshi

(National Key Labotayory of Science and Technology on Multi-spectral Information Processing, School of Automation, Huazhong University of Science and Technology, Wuhan 437704, China)

Abstract: In this paper,a method to achieve timing closure during synthesis and physical design stage of digita IC is discussed. Timing closure is one of the most important task in digital integrated circuit design. As integrated circuit develops into the deep sub-micron period, the size of chip and the complexity of design has increased significantly. As a consequence of this, it becomes more and more difficult to achieve timing closure. This paper has discussed the method to achieve timing closure in synthesis and physical design based on low-level image processing SoC design. This paper firstly introduces the way that Design Compiler and IC Compiler use to perform timing analysis, followed by a discussion of how to use the tool to set reasonable constraints on the design, and finally introduces the solutions to timing violations in various stages. This SoC has finally got timing closure by using this method..

Key words: integrated circuit; timing; synthesis; physical design

0 引言

时序是保证数字电路正确工作最关键的因素。随着集成电路制造工艺的进步，芯片设计日趋复杂，为达成时序收敛，设计者也面临越来越大的挑战[1]。图像中低层处理 SoC 是本实验室设计的一款 SoC。芯片采用 SMIC 0.18μm 工艺，最高时钟频率达到 100MHz。图像中低层处理 SoC 的后端设计流程中，需要考虑时序收敛的包括综合、布局规划、时钟树综合、布线等阶段，不同阶段有着不同的处理方式[2]。本文基于此 SoC 的设计过程，结合本人数年来的数字集成电路设计经验，提出了一种数字集成电路设计中，综合以及布局布线阶段的时序收敛方案。该方案首先对芯片做出合理而详尽的约束，包括综合阶段的严格约束、布局布线阶段改变约束使其贴近实际情况以及使用布局布线过程中独有的约束方式如多场

基金项目：总装预研教育部支撑计划重点项目“微型 XXXX 光电子系统集成技术”(625010107)

作者简介：谢扬（1989-），男，硕士，主要研究方向：数字集成电路后端设计

通信联系人：桑红石（1970-），女，副教授，主要研究方向为图像处理和集成电路设计. E-mail: ellen_shs@qq.com

景分析、OCV 分析、CTS 约束、时钟网络布线约束和串扰分析等方法。在此基础上得到一个比较好的结果，再针对各阶段出现的部分违例进行针对性的优化，最终使系统时序收敛。

1 时序分析基本原理

传统的静态时序分析将整个设计打散为一条条路径，根据其起点和终点的不同主要分为以下四种形式：从输入端口到触发器数据端、从触发器时钟端到触发器数据端、从触发器时钟端到输出端口和从输入到输出端口[3]。

随后，工具对每一条路径进行检查。时序检查的对象有很多，其中最重要的是建立时间（setup time）和保持时间(hold time)检查。如图 1 所示，建立时间是指在时钟有效沿到来之前，数据必须保持稳定的时间；保持时间是指在时钟有效沿到来之后，数据必须保持稳定的时间。本文主要针对这两项检查展开讨论。

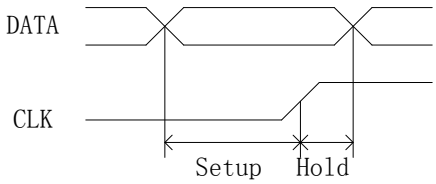

图 1 建立/保持时间的定义

Fig. 1 Concept of Setup Time and Hold Time

2 各阶段的时序约束

EDA 工具对时序的分析和优化是由约束来驱动的，为保证设计的时序，首先就要进行时序约束。时序约束的方式多种多样，应根据设计的具体情况，进行详尽而合理的约束。此外，在数字集成电路设计的各个阶段，应该随时调整约束，以达到最好的效果。下文从时序约束的基本方式、使用 Design Compiler 综合时的约束方式和使用 IC Complier 物理设计时的约束方式等三方面进行讨论。

2.1 时序约束的基本方式

时序约束最基本的内容包括时钟约束、输入输出延时约束和对芯片边界环境的约束[4]。时钟约束是整个时序约束中最重要的部分之一，它影响着所有由它驱动的路径的时序分析。完整的时钟约束应包括周期（period）、延迟(latency)、转换时间（transition）和不确定性（uncertainty）[4]。周期是时钟最基本的参数。延迟包括时钟源延迟（source latency）和时钟网络延迟（network latency）。其中，source latency 即时钟源到当前芯片或模块的时钟根节点的延迟，nerwork latency 是时钟根节点到叶节点（触发器或者宏单元时钟端）的延迟。转换时间体现时钟信号翻转过程的快慢。不确定性包括时钟偏差（skew）、时钟抖动（jitter）和裕量（margin）三个部分。其中,skew 是同一时钟到达该时钟域内不同叶节点之间的偏差，jitter 是时钟实际到达时间与理想到达时间之间的偏差，margin 是设计者为增加系统适应性而人为添加的裕量。

完成对时钟的约束，即可约束上述四种类型路径中的第二种，即内部路径。相对于经典的两级触发器时序分析模型，另外三种路径实际上是不完整的，因此需对其设定输入延时（input delay）和输出延时（output delay）的约束。这两者分别表征从当前模块算起，输入路径和输出路径上的延迟。

最后，需对芯片边界的环境进行约束。EDA 工具计算标准单元的延时是基于输入转换时间（input transition）和输出负载(output load)。因此需对芯片的输入端口设置驱动，对输出端口设置负载。

2.2 综合时的约束方式

图像中低层处理 SoC 采用 Design Compiler 进行逻辑综合。设计中包含四个外部输入时钟、两个片上分频时钟以及三个输出时钟（表 1）。

表 1 图像中低层处理 SoC 时钟说明

Tab. 1 Clock Definition of Low-level Image Processing SoC

时钟名	频率/MHz	说明
rpad_clk	100	芯片主时钟
rpad_clk_ADC_10	10	ADC 采样时钟
rpad_clk_33_image	50	中间结果输入同步时钟
TAP_TCK	10	JTAG 时钟
clk_fpa_10	10	fpa 模块时钟，由主时钟十分频而来
clk_o_10	10	液晶控制模块时钟，由主时钟十分频而来
rpad_EMIF_clk	100	处理数据输出时钟、sdram 访问时钟
rpad_clk_o_led	10	8 路方波输出参考时钟
rpad_clk_o_fpa	10	fpa 输出参考时钟

由于综合时与布局布线时的时序收敛难度差异很大，因此选择综合时普遍采取比较紧的约束，以达到与后续工具更好的协同效果。首先，约束时钟时，时钟频率均设定为原始频率的 120%。由于综合时 DC 计算 AT 与 RT 时均算了 latency 值，所以在综合中一般可以不设置 latency，除非已知不同时钟带有不同的 source latency，并且它们之间有时序要求。或者已知不同时钟有不同的 network latency，不想平衡它们，但是要满足他们之间的时序要求。Transition 的设置与工艺相关，本设计采用 0.18μm 工艺，根据经验值设为 0.3ns。uncertainty 在综合中可以考虑为 PLL jitter、估计的 skew、margin 三个部分，后两者根据经验设为时钟周期的 5%。除此之外，综合过程中由于没有时钟树，因此需要将时钟设为理想网络，以阻止工具对其进行 DRC 检查和针对违例进行修改。复位等典型大扇出网络同样应该做此处理。对于表 1 中的生成时钟，还需要指定其主时钟和分频比例。对于输出时钟，为了保证与之相匹配输出数据的同步性，不建议对其进行简单的定义。本设计中采取的方法是将其定义为驱动该数据输出时钟的一分频生成时钟，然后将与之对应的输出端口的输出延时参考它设定，最后经过综合以及时钟树综合（CTS）时的处理，可以实现更好的同步。此外，本设计中时钟较多，为使工具正确分析有逻辑关系的同步时钟域内的路径，应根据设计来确定各个时钟之间的关系，然后通过 set_clock_groups -asynchronous 来约束它们之间是同步还是异步。

其次，约束输入输出端口时，主要采取根据经验估计加上查看外设 datasheet 两者结合的方式进行。总体原则仍是尽量做紧的约束，如无论是输入还是输入延时，均在片外部分分配周期的 60%~70%，然后结合外设（本设计中表现为一块 sdram 和一块 flash）的时序参数进行调整。若初步综合完的结果中在边界路径上出现部分微小违例，可以适当放宽约束。此外，本设计中还存在一些与时钟无关的电平信号，对它们进行时序分析是无意义的。对这些信号可以不做输入输出延时约束，而是通过以下命令将其设置为 false path，以禁止工具去分析其时序。

最后，约束环境时，对内核进行综合时可将相应端口的驱动和负载设为工艺库 pad 的对应端口。对带 pad 的顶层进行综合时，应根据板级情况，结合经验进行考虑，设为一个定值。

此外，本设计还采用了 DCT 的流程进行综合。传统的综合没有任何物理信息，通过估算互连线的长度来进行时序分析。DCT 流程需要拿初步综合的结果进行布局布线并吐出一个包含布局信息的文件（.def），再让 DC 基于此文件进行综合。将这个过程不断迭代以达到最优的时序情况并做到与布局布线最好的一致性。基于以上考虑，最终综合结果达成了时序收敛，并且得到的是一个裕量比较大的设计，使后续工作能更方便地进行。

2.3 物理设计时的约束方式

相对于综合阶段，物理设计阶段的特点是有更明确的物理信息、时序分析情况更复杂、分析结果更精确。物理设计大致可分为布局规划、CTS、布线和验证四个阶段[5]。图像中低层处理 SoC 使用 IC Compiler 进行物理设计，用到了全阶段采用多场景分析和 OCV 分析、CTS 阶段改变约束并优先布线时钟网络、全阶段考虑串扰等方法。

首先，本设计采用了多场景分析和 OCV 分析。在实际应用中，芯片工作的模式会存在多种情况，常见的有功能模式、扫描链测试模式和边界扫描模式等。此外，芯片工作的环境也可能发生变化，IC Compiler 将其归纳为三个因素，即 PVT。其中 P 为制程（process），V 为电压（voltage），T 为温度（temperature）。在不同的模式或不同的环境下工作时，芯片的时序会发生变化。为保证芯片在各种情况下均满足时序，IC Complier 提供了 MCMM 的流程。此流程通过设定不同的场景（scenario），再分别对它们进行约束和优化，以满足各场景的时序要求。每个场景由 PVT、工作模式和寄生参数文件（TLUP）组成。在定义场景时，除了申明以上三要素，还需要读入对应该场景的时序约束文件（.sdc），并且将相应的控制端口（如测试模式控制端口）置为常值。本设计还采用了 OCV 分析。芯片正常工作时，所谓“一处是火山，一处是海洋”，片上各处环境不全然相同。为了更精确的模拟实际情况，要考虑不同时序路径上的具体情形。对于图 2 中的路径，传统的 bc_wc 分析方式在计算 setup 时，launch path 和 capture path 均按最大值（max）进行计算，即传播最慢的 slew 值；在计算 hold 时，两者均按最小值（min）进行计算，即传播最快的 slew 值。这样的分析方式在 capture path 上显得乐观。本设计中采取 OCV（On Chip Variation）的方式进行分析，在计算 setup 时，launch path 按 max 进行计算，即传播最慢的 slew 值；capture path 按 min 进行计算，即传播最快的 slew 值。计算 hold 时则相反。相比 bc_wc 方式，显然这种方式更加严格。OCV 通常与 MCMM 结合使用。在设定场景的时候，可以将分析方式设定为 OCV，并指定 max 和 min 对应的逻辑库。表 2(a)说明了厂家为每个工艺角提供单个库文件时的，在最坏工艺角下不同分析类型与逻辑库的对应关系；表 2(b)则是厂家为每个工艺角提供多个库文件的情形。本设计每个工艺角下只有单个库文件，为进一步加强约束，采取了 ICC 提供的一种 timing derate 的方式来模拟表 2(b)中的情景，如在 set_timing_derate -min -early 0.95 约束下计算 setup 时，需在 capture path 实际的延时上乘以 0.95；计算 hold 时，则是在 launch path 实际的延时上乘以 0.95。这无疑是一种非常严格的约束。最终，本设计根据实际情况分为表 2(c)所示四个场景。随后在布局、CTS 及布线过程中对四个场景的所有时序路径进行检查和优化，并使芯片满足了所有场景的时序要求。

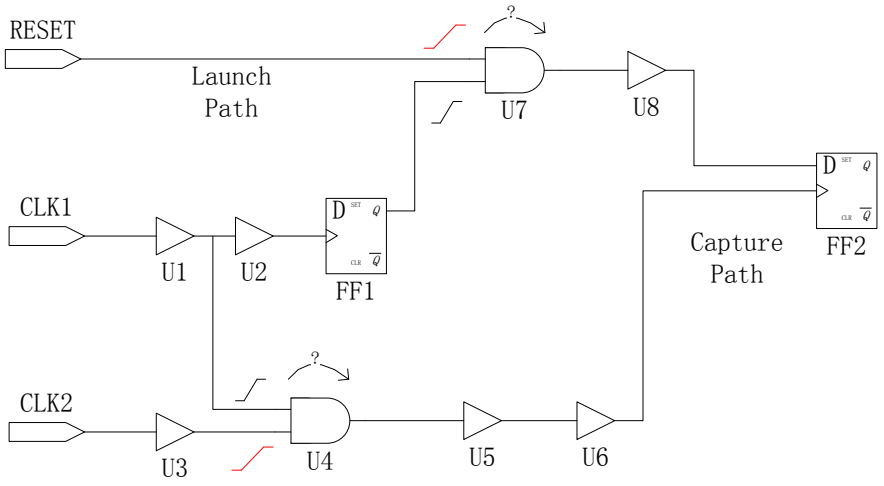

图 2 OCV 分析方式

Fig. 2 Concept of OCV

表 2 MCMM 与 OCV 分析

Tab. 2 Concept of MCMM and OCV

(a) 图像中低层处理 SoC 场景设定

场景名	PVT	模式	TLUP
func_worst_corner	Worst	function mode	Max
func_best_corner	Worst	function mode	Min
test_typical_corner	Typical	scan chain test mode	Typical
JTAG_typical_corner	Typical	JTAG mode	Typical

(b) 单个库文件工艺角与逻辑库对应关系

	Setup check	Hold check
Launch path	-max WORST	-min WORST
Capture path	-min WORST	-max WORST

(c) 多个库文件工艺角与逻辑库对应关系

	Setup check	Hold check
Launch path	-max WORST	-min WORST_ocv_min
Capture path	-min WORST_ocv_min	-max WORST

其次，在 CTS 阶段改变了约束，采用了迭代方式以做出最佳时钟树。CTS 即时钟树综合，是数字后端中最重要的环节之一。在 CTS 之前，系统不存在时钟树，单个时钟端口驱动成千上万个寄存器，这显然是不符合物理上的要求的。因此，设计者需要通过完成 CTS 来构建一个完整的时钟树，使得每个时序单元都能被成功地驱动，并且要把时钟的 skew 和 insertion delay 控制在一定范围内，以达到系统的时序要求。相比综合阶段，此时的基本约束应该随阶段不同而改变。改变主要体现在时钟的相关参数上，需要进行以下处理：（1）去掉所有时钟的理想网络属性；（2）将所有时钟设置为传播时钟，让工具计算其 network latency：通过以上命令可以计算出实际路径上时钟的 insertion delay，对于一部分虚拟路径，如边界上不完整的路径，还需要在 CTS 的主命令 clock_opt 中加上选项-update_clock_latency，以计算这一部分的 insertion delay；（3）将所有时钟的 uncertainty 中的 skew 部分去掉，此时时钟的 skew 已体现在第二步的计算中，若仍然包含 skew 则会得到过于悲观的结果；（4）设定 CTS 目标，包括 target skew 值和 min insertion delay 工具会在满足 target skew 的情况下再去满足 min insertion delay。这两个值要不断地尝试，通过生成时钟树再报出相关参数的迭代过程，找出工具能做出最好时钟树的情况；（5）对于系统的输出时钟，在 CTS 过程中是

默认为 exclude pin，IC Compiler 不会对其进行 skew 的平衡。为保证输出数据与时钟的同步性，在前文已提到将其输出延迟参考输出时钟定义，在此还需将时钟定义端口设为 stop pin，以让工具去平衡 skew[6]。CTS 完成后，应让时钟网络先于普通信号线进行布线，这有利于使其占有更有利的布线资源，还应选择高层金属进行时钟网络的布线，因为高层金属的 RC 值更小，这都有利于获得更好的时序。

最后，图像中低层处理 SoC 设计全阶段都考虑了串扰（Crosstalk）的影响。串扰体现为两条距离很近的金属线，一方（aggressor）发生翻转时会通过两者间的耦合电容对另外一方（victim）产生影响。当 victim 处于静止状态时，此影响体现为毛刺；当 victim 同时也在翻转时，则取决于两者翻转方向的异同，其翻转时间会增加或减小。显然这会影响系统的时序，甚至导致错误的逻辑被传播。

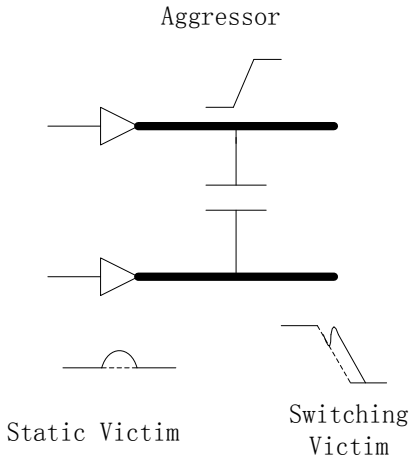

图 3 串扰的影响

Fig. 3 Effection of Crosstalk

随着集成电路制造工艺的进步，串扰对时序的影响也变得越来越严重，甚至可能造成时序检查通过，然而实际工作时无法满足时序要求的情况。对于本设计采用的 0.18μm 工艺，为增加芯片的抗串扰能力乃至保证其时序，需要进行串扰分析并在布线过程中考虑其影响，对其进行优化。IC Compiler 对串扰的处理主要体现在两个方面，即布线前预防和布线后优化。布线前预防包括：（1）在布局时通过设定 set_congestion_options 以及 area_recovery_critical_range 和 power_recovery_critical_range 等约束来控制宏单元的位置、拥塞（congestion）等因素，调整单元的位置以减少网络交互的距离和间距。（2）使用 set_max_transition 约束最大输入转换时间（input transition）。Crosstalk 本质上是一个电容值，通过约束每条网络的 input transition 即驱动强度，可以有效降低其影响；（3）在关键路径上使用自定义的规则，如在时钟网络布线时采用的双倍宽度双倍间距规则；（4）在布线过程中通过 set_si_options 打开防止 crosstalk 的选项开关。另一方面，要进行布线后优化，首先需要用命令 set_delay_calculation -arnoldi 将工具的时序分析模型从传统的 Elmore 模型转换为考虑串扰影响的 Arnoldi 模型。完成这个转变后去分析时序，肯定会得到一个相对较差的结果，这也说明了串扰对时序的影响。随后即可用命令 route_opt -xtalk_reduction 针对串扰进行优化。

3 解决时序违例的方式

3.1 各阶段关注的重点

由于从综合到布局布线乃至签核整个流程的特性，在各个阶段需关注的时序分析重点也各不相同，如表 3 所示。

表 3各阶段关注的时序重点

Tab. 3 Key Point of Timing Analysis in Different Stage

阶段	Setup	Hold
综合	检查且修复	不关心
布局布线（CTS 前）	检查且修复	不关心
布局布线（CTS 后）	检查且修复	检查，可以不修复
布局布线（布线后）	检查且修复	检查且修复

3.2 修复时序违例

时序违例在各阶段都会出现，各阶段也都有自己偏重的修复方法。下文将介绍综合、CTS 前后和布线后修复时序违例的方法，并以本设计中功能模式下的系统主时钟域内的时序情况为例，展示利用这些方法得到的结果。

首先，在综合阶段，一方面代码尚未完全定稿，另一方面在整个芯片设计中，利用种种约束和优化方式是必需，但所有这些方法对于系统性能的提升都是有限的，系统的性能最终还是取决于结构的设计（architecture）和编码风格（coding style）。因此主要修复违例的方式还是基于修改代码。另外，从本文第二节的分析可以看出，布局布线阶段的时序分析比综合时复杂的多，因此综合时需要把握的原则是下足够紧的约束，吐出有足够裕量的设计，以方便后续工作的开展。这个阶段是一个不断迭代的过程。

以本设计为例，拿到第一版代码后应该做一个粗略的约束，通过 DC 报出的警告甚至错误来修改和完善代码。随后在完善的时序约束下乃至利用 DCT 进行综合，通过 report_qor、report_timing 等命令找出整个设计里的所有违例情况和关键路径，集中优化这些路径。具体可以采用优化代码结构、优化运算电路逻辑、指定 designware 使用时序性能更好的单元、流水线设计、多周期路径设计等方法。本设计的综合结果体现了很多问题，如不完整的条件判断语句综合生成了锁存器，可以通过优化代码解决。又如某模块用到了大量卷积运算，最初将其放在一个周期内完成，这显然是不现实的。后来将其拆分成若干级流水结构，并将其设计为多周期路径，达到了时序要求，如表 4(a)所示。还有一部分违例路径包含一些大扇出网络，通过 report_timing -capacitance 可以报出这些路径上的负载情况，若存在大扇出，可以通过修改代码结构或者在综合时 set_max_fanout 来施加扇出约束，同样可以有效的改善时序。

其次，在 CTS 阶段前，修复时序违例重点在于调整布局。布局是整个数字后端最重要的环节，一个好的布局可以为系统的时序、功耗、IRDROP 等多方面的性能打下良好的基础。若此阶段出现时序违例，应着重从两方面调整布局，即 PAD 的摆放和宏单元的摆放。PAD 根据其功能特性可分为电源 PAD、输入信号 PAD、输出信号 PAD 等种类。摆放 PAD 的原则之一是尽量将数据流相同的 PAD 放在一起，例如将所有输入信号 PAD 放在一起，所有的输出信号 PAD 放在一起，若让两者交错放置，会延长数据流路径，从而影响时序。此外，还要从板级考虑每个 PAD 的位置，将它们放在便于与板级其他设备交互的地方。硬宏单元是设计模块中个体面积最大的单元，是连接 PAD、标准单元的枢纽。硬宏单元占据了 Soc

中相当大一部分面积，如果硬宏单元摆放不合理，将会引起标准单元拥挤，增大连线长度，从而影响时序。硬宏单元一般摆放在整个芯片四周，为标准单元留出中间的空位，并且要结合芯片内部结构的数据流以及 PAD 的位置决定其具体位置。本设计中存在 200 个 PAD 和 59 个硬宏单元，经过调整布局后如图 4 所示，时序如表 4(b)所示。

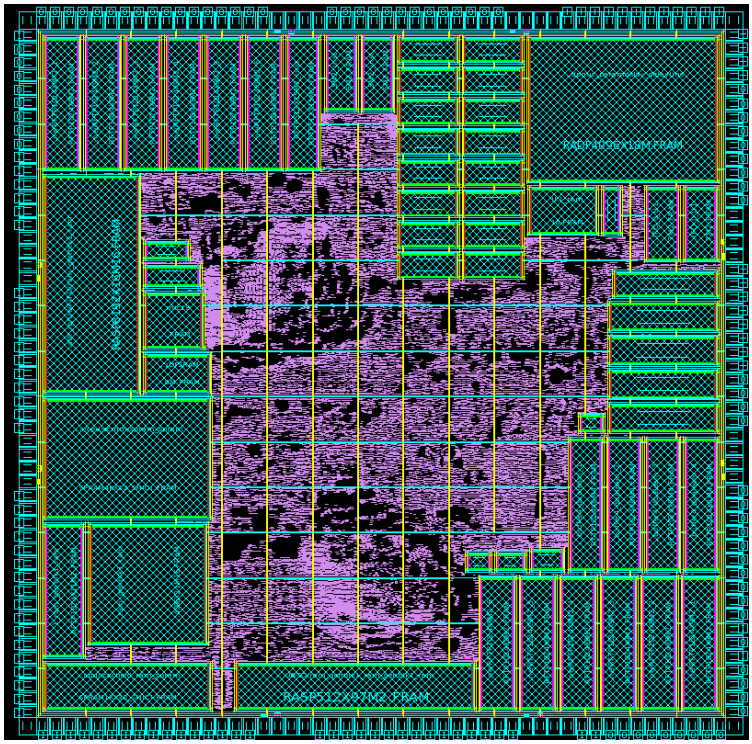

图 4 图像中低层处理 SoC 最终布局

Fig. 4 Final Placement of Low-level Image Processing SoC

第三，在 CTS 完成后若出现违例，在使用本文 3.3 节提到的技巧的同时，一方面要多调整 CTS 的参数，以做出更好、更平衡的时钟树；另一方面还可以手动调整 buffer/inverter 的尺寸，避免其太大或者太小，太大的容易引起 EM 问题，并且在做 crosstalk 分析时有很强的的侵略性；而太小的驱动能力不够，用在时钟网络这样关键的路径上是不合适的。本设计中经过数次迭代与修改，迭代前与最终迭代结果如表 4(c)所示。

最后，完成布线后，若出现违例，此时调整的空间相对前面环节已经不大。主要通过微调时钟树中 buffer 的尺寸，优化串扰以及利用 ECO 解决少量微小违例，优化串扰前后对比如表 4(d)所示。如果违例情形比较严重，则需要返回到 CTS 甚至布局阶段。总而言之，由于布局布线中阶段较多，分析方式趋于复杂，一定要尽早发现问题、解决问题。设计过程中敢于尝试，以获得最好的结果。

表 4 各阶段时序违例解决前后对比

Tab. 4 Comparison of Timing Before and After Optimization in Different Stage

(a) 综合中对卷积运算电路多周期处理前后时序对比

参数	单周期	双周期
Slack	-7.3ns	0.8ns

(b) 默认布局与调整后布局时序对比

参数	默认布局	调整后布局
Slack(func_worst_corner)	-0.332ns	0ns
Slack(func_best_corner)	0ns	0ns

(c) CTS 调整前后时钟树参数对比

参数	迭代前	迭代后
Max Skew	1.241ns	0.813ns
Max Insertion delay	4.731ns	4.269ns

(d) 布线后优化串扰前后时序对比

场景	优化前 Setup	优化前 Hold	优化后 Setup	优化后 Hold
func_worst_corner	-0.331ns	-0.201ns	0ns	0ns
func_best_corner	0ns	-0.125ns	0ns	0ns

4 结论

本文以图像中低层处理 SoC 为例，结合作者数年来设计经验，阐述了基于标准单元的数字集成电路设计流程中，从综合到物理设计的时序收敛方案。最终结果表明本设计达到了实现收敛，证明了本文论述的有效性。

[参考文献] (References)

[1] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic．数字集成电路--电路、系统与设计[M]．周润德等．北京：电子工业出版社，2010
[2] 张志．NUCSoC 芯片的物理设计[D]．武汉：华中科技大学，2011
[3] Synopsys．Design Compiler Workshop[M]．USA：Synopsys，2008
[4] 陈春章，艾霞，王国雄．数字集成电路物理设计[M]．北京：科学出版社，2008．
[5] Synopsys．IC Compiler 1 Workshop[M]．USA：Synopsys，2008
[6] 千路，林平分．ASIC 后端设计中的时钟偏移以及时钟树综合[J]．半导体技术，2008，33(6)：527-529．