怎样能更好地应用工具进行 RTL 综合研究

山 霞[1],田 媛[2]

(1.上海电力学院,上海 201906;
2.上海交通大学,上海 200040)

摘 要:针对当前 RTL 综合面对的挑战,总结了实际项目中的经验,可以使综合工具在更少的时间里产生的网表芯片面积更小、速度更快,而功耗更低。

关键词:RTL 综合;CMOS 电路;设计自动化

中图分类号:TN402 **文献标识码**:A

Research on how to use tool to do RTL synthesis

SHAN Xia[1], TIAN Yuan[2]

(1.Shanghai University of Electric Power, Shanghai 201906, China;
2.Shanghai Jiaotong University, Shanghai 200040, China)

Abstract: This paper discusses some rules that have been deployed for RTL synthesis that significantly improves throughput. In particular shows, the techniques presented in this paper get faster, smaller and cooler chips in less time.

Key words: RTL synthesis; CMOS integrated circuits; design automation

随着微电子设计复杂度不断增加,生产工艺的不断缩小,对设计自动化工具提出了不断的挑战。电子设计自动化需要能处理非常复杂的设计所产生的挑战和需求。逻辑综合更是应该满足这一需求。逻辑综合工具应该可以在合理的时间里处理上百万甚至千万门的逻辑电路。而传统的逻辑综合理论主要是靠电路综合转换的反复试验,这个过程其实是功效和质量的折中,通常不能产生最优化的结果。而且,产生的电路非常依赖于具体的设计。因此,只有非常有经验的设计工程师才能综合出优化的电路。面对这些挑战,可以用本文所阐述的方法,在更少的时间里,综合出速度更快、面积更小、功耗更低的芯片。其技术是基于新出现的综合工具,不断增强的后端工具,更好的计算机资源和更快的操作系统。通过应用这些技术,已经成功地完成了一个 90 nm 芯片项目,特别是使用这些技术产生的网表,可以在后端实现的时候很快地时序收敛,保证了芯片的按时完成。

怎样指导综合工具产生想要的电路?当使用工具综合设计的时候,首先明确的是工具永远按照工程师的指导在工作,这就需要工程师在做综合之前,首先要清楚需要完成什么样的电路。比如电路的工作频率,有无低功耗要求,面积要求等;其次要清楚工具会为我们做什么,而不能将工具看成一个万能的工具;然后要指导约束工具综合产生出我们需要的电路。接下来介绍应用哪些规则和方法,可以综合产生更好的电路。

1 重视 RTL 编码设计

第一个规则就是注重 RTL 设计。无论综合工具提高了多少,多么好的计算机和多快的操作系统,都要从设计刚开始的时候就注重 RTL 编码设计。综合是将设计理念从一种表达形式(RTL 功能描述),转换成另一种形式(针对于某一工艺的门级网表),电路在综合前后具有相同的功能[1]。不能将综合看成一个黑盒子而不知道里面的内容。必须清楚的知道所需要实现的电路、设计的特殊性、应用的综合工具、还有工具如何处理逻辑电路。不要依赖后端工具来解决时序问题,而是要在开始设计编码的时候,就应考虑怎么解决这些问题。在项目开始的时候,应用从先前芯片设计中得到的经验,结构化设计理念,更有可能产生高效的电路。预先计划怎样描述

RTL 设计，就不用在后端实现的时候花很多时间进行布局布线的设计。

2 采用自上而下的综合

综合工具的性能已经提升了很多，能够在可以接受的时间里完成自上而下(top－down)的综合，而不用从下而上(bottom－up)的综合。这样可以简化综合约束的管理，减少编译设计所需的综合脚本，也就减少了使用综合工具所需的许可证。自上而下的综合可以使工具在优化时知道设计在模块间的拓扑连接、负载、时序，从而能综合产生更好的电路。使用自上而下的综合方法，还可以消除综合对时序预算的依赖。

3 只对关键路径加额外的约束

传统的综合流程里，为了能在后端实现时留有一定的余地，通常建议在加约束的时候，将时钟额外增加 10%，有些设计甚至增加更多，或者用更大的连线模型，以留有足够的余地。一些设计在加约束时，增加了如此多的额外约束[3]，以至于虽然设计在后端实现的时候满足了时序收敛的要求，但是却以牺牲面积为代价，而且有些可能是不必要的。

现在的工具可以明确需要在哪些关键路径上加额外的约束，以使这些路径在后端实现时能够满足时序的要求，而不用将整个时钟都加额外的约束。如果将时钟加额外的约束，将会使设计中使用这个时钟的电路都被不必要的优化，包括那些在后端实现的时候时序已经收敛的电路。

在本方法的流程里，定义了实际需要的时钟，不是将整个时钟加额外的约束，而是只针对关键路径加额外约束，如图 1 所示。

不可否认，这个流程仍然需要反复，但是通过这个流程综合产生的网表能够得到更好的时序收敛结果。因为在后端实现中的关键路径在综合时被加了更快的约束，而非关键路径没有加额外约束，从而减少了芯片面积，降低了成本。

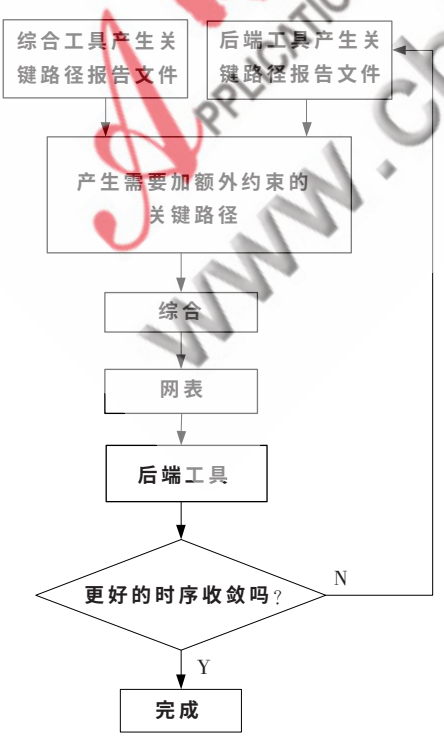

图 1 对关键路径加额外的约束

4 重视寄存器与寄存器之间路径的最优化

通常，设计中的输入输出(I/O)成为整个芯片工作频率的瓶颈，而核心的逻辑能在更快的频率上工作。所以在综合时，放松加在 I/O 上的约束，使综合工具能够专注在寄存器到寄存器路径的优化。待路径优化满足了设计的需要，再关注输入输出路径，或者对输入输出路径做一些结构的微调以满足产品的需求。通过这种方法，可以使综合工具先优化寄存器到寄存器的路径，然后再优化输入输出路径，达到设计要求。

5 利用电容及电阻的物理信息

传统的基于输出连线负载模型不能很好地模拟连线间的延迟。如果设计的大小在优化时发生了变化，即使是定制的连线负载模型也是低效的，因为它是“静态”的。现在的综合工具可以读入物理库文件，可以更好地模拟，并动态地调整物理连线上的时序。例如，Cadence 公司的 RTL Compiler 在综合的时候，利用布局评估器(PLE)使用物理信息来计算线延时[2]。虽然 PLE 没有芯片的物理布局布线信息，但是可以用读入的 LEF 文件，或者电容文件估算出线延时。在综合时，利用物理信息来估算连线的电容、电阻以计算线延时比用传统的连线负载模型更准确、更高效，而且也容易使用。

6 设置正确的时序特例

时序特例(timing exception)是综合工具默认的单周期时序行为的特例。对于设计中每一个不符合默认时序特征的电路，都要设置声明时序特例，以使综合工具不去优化这条电路，或用更宽松的约束去优化。

时序特例共 3 种：电路延迟、多周期电路和虚假电路。多周期电路是指需要超过 1 个时钟的时间来传送数据，需要对这个传送间隔声明 1 个有限个数的时钟周期。电路延迟特例是指电路的延迟需要满足具体说明的延迟[4]。电路存在于设计中，但是不会被激活，称为功能虚假电路。综合工具会优化这些功能虚假电路以满足时序的要求。综合工具是基于静态时序分析的工具。静态时序是基于连接关系，而不是功能。不管这些路径是否会在芯片中真的用到，时序分析都会基于连接关系显示出这些关键路径。可以通过声明路径是功能虚假电路来消除很多不需要的关键路径。越早清除这些虚假路径，综合工具就能越早地优化那些真正的路径，也就能更有效地达到时序收敛。所以，设置正确的、有效的时序特例至关重要。

7 允许新的时序优化

新的时序优化技术可以综合产生出更好的电路，但是通常会有逻辑等效性检查的问题。为了避免逻辑等效性检查题，应尽量使用最新版本的逻辑等效性检查工具，或者使用与综合工具相对应版本的逻辑等效性检查工具。逻辑等效性检查工具在理解综合优化上已经取得了很大的进步。逻辑检查工具现在可以验证很多综合优

化，比如寄存器重新调整(register retiming)、边界优化、数据分支优化和时序优化等。不要因为这些新的时序优化技术可能产生的逻辑等效性检查的问题就关掉这些优化，这样会影响综合出的电路性能。

8 不要限制对库单元的选择

在传统的综合中，为了得到更好的网表结构，往往限制对库单元(library cell)的选择。而现在的综合工具是基于所加的约束条件和目标单元库来产生逻辑电路。因此，可供选择的库单元越丰富，综合工具就越可能优化电路，从而平衡电路的面积、速度和功耗，可以给后端提供更好的电路网表。

本文阐述了多种应用在SoC芯片设计中的综合技术，应用这些技术和规则，可以更加充分地利用现代综合工具和软硬件环境，从而能在更短的时间内，使综合产生的网表电路速度更快、面积更小、功耗更低，后端实现时也能更快地进行时序收敛，从而满足芯片设计的要求。今后将继续优化本文介绍的综合技术，跟踪使用新的综合工具，以优化本文的芯片设计。

参考文献

[1] Synopsys Inc. Design compiler reference manual. Release 2006.06, Mountain View, CA, 2006.

[2] Cadence Inc. RTL compiler reference manual. Release 6.2.2, Seely Ave., San Jose, CA, 2007.

[3] YU Xin You, HONG Peng, YAN Hui Yang. Signal integrity timing closure of million gates SOC platform. IEEE ASICON, 2005.

[4] JUNG J Y, KIM T. Timing variation-aware high-level synthesis IEEE Computer-Aided Design, 2007.

(收稿日期：2009-03-13)