A Local Optimization Method for Single Flux Quantum Logic Circuits Design Utilizing Synchronizer IPs

Intro

- 使用同步器IPs, 针对逻辑电路设计的局部优化方法
- 属于EDA工具的开发

The significant feature in SFQ circuits design using EDA tools typically requires path balancing, which means D-Flip-Flops (DFFs) should be inserted to ensure the same logic level of all gates in the same stage. Path balancing in SFQ design ensures that all input signals of a gate have the same clock period, but the latter can also be scaled and automated in logic synthesis in the developed EDA tools and maintain high throughput characteristics of SFQ circuits. SFQ设计中的路径平衡可确保门的所有输入信号具有相同的时钟周期，但后者也可以在开发的EDA工具中进行逻辑合成的缩放和自动化，并保持SFQ电路的高吞吐量特性。

This article presents a local optimization method that addresses the excessive consumption problem of the clock tree for full path balancing in SFQ circuits. The suggested scheme primarily utilizes the EDA tools JSICcomplier for SFQ circuits, which is self-developed by our team. Specifically, we design synchronizer IPs called Dsyn series to replace DFF for path balancing. In this way, clock nodes and the clocking path consumption are reduced, minimizing changes in the JSICcomplier tools algorithm. The suggested method is compatible with a switch in our existing tools, offering a choice in optimization in clock tree synthesis and logic synthesis.

The rest of this article is organized as follows. Section II briefly introduces the JSICcomplier tools for SFQ circuits developed by our team. Moreover, it proposes an S-box circuit design of a masked AES-128 algorithm under our EDA tools as a demo to show the enormous path balancing consumption. Section III provides our local optimization method on path balancing involving different test groups to demonstrate our method's effectiveness in each group. Section IV gives the experimental results and analysis under several circuits. Finally, Section V concludes this article.

- 本文提出了一种局部优化方法，用于解决SFQ电路中全路径平衡时钟树的过度消耗问题。提出的方案主要利用我们团队自主开发的用于SFQ电路的EDA工具JSICcomplier。
- 设计了名为Dsyn系列同步器IPs替代DFF进行路径平衡

Concepts

1. path balancing 路径平衡 为了确保电路在每个阶段都正常工作, 我们需要通过插入DFF来平衡或对齐所有逻辑门的输出. 因为在高频和高吞吐量的SFQ电路中, 任何小的时间差异或逻辑不一致都会导致电路的错误.

   通过使用DFF, 可以确保所有逻辑门在相同的时间点上具有相同的逻辑状态, 从而使整个电路更加稳定

   例如，想象一个简单的逻辑电路，其中有两个逻辑门A和B。如果A的输出比B稍微快一点，那么下一个阶段的电路可能会因为A和B的输出不同步而出错。通过插入D-Flip-Flops，我们可以确保A和B的输出在相同的时间点变得同步，从而避免这种不一致。

   gates in most gates are different. To reduce additional resource consumption, several path balancing mapping algorithms have been proposed [9], [10], aiming to minimize the depth and path balancing overhead. An alternative method inserts a DFF or a part of DFFs in the SFQ circuits using a new pipelined architecture with dual clocks [11], despite degrading the peak throughput of the circuit. Considering full path balancing, the consumption originates from the extra gates (DFFs) and the routing involving a much larger clocking tree than the one prior to path balancing. In SFQ digital circuits, the clock distribution network is more complex than CMOS for each clock-driven. Therefore, the SQF clock tree is much larger than that of CMOS, and from the perspective of resources, designing the clock tree needs more attention.

2. path balancing mapping algorithms路径平衡映射算法 一种使用新的流水线双时钟架构在SFQ电路中插入DFF或部分DFFs的方法。这种方法可能会降低电路的峰值吞吐量。 在SFQ数字电路中，时钟分布网络比每个时钟驱动的CMOS更复杂。因此，SFQ的时钟树比CMOS的大得多。

Circuits design Using JSICcompiler

A. Brief Introduction of EDA Tool JSICcomplier介绍一种EDA工具 JSICcomplier是一款EDA软件，用于SFQ逻辑电路的逻辑合成、放置、布线和优化。其中，逻辑合成部分需要其他EDA工具的帮助进行Verilog编译，而完整的

SFQ逻辑编译模块将在近期添加。图 1 说明了 JSICcomplier 用户界面。

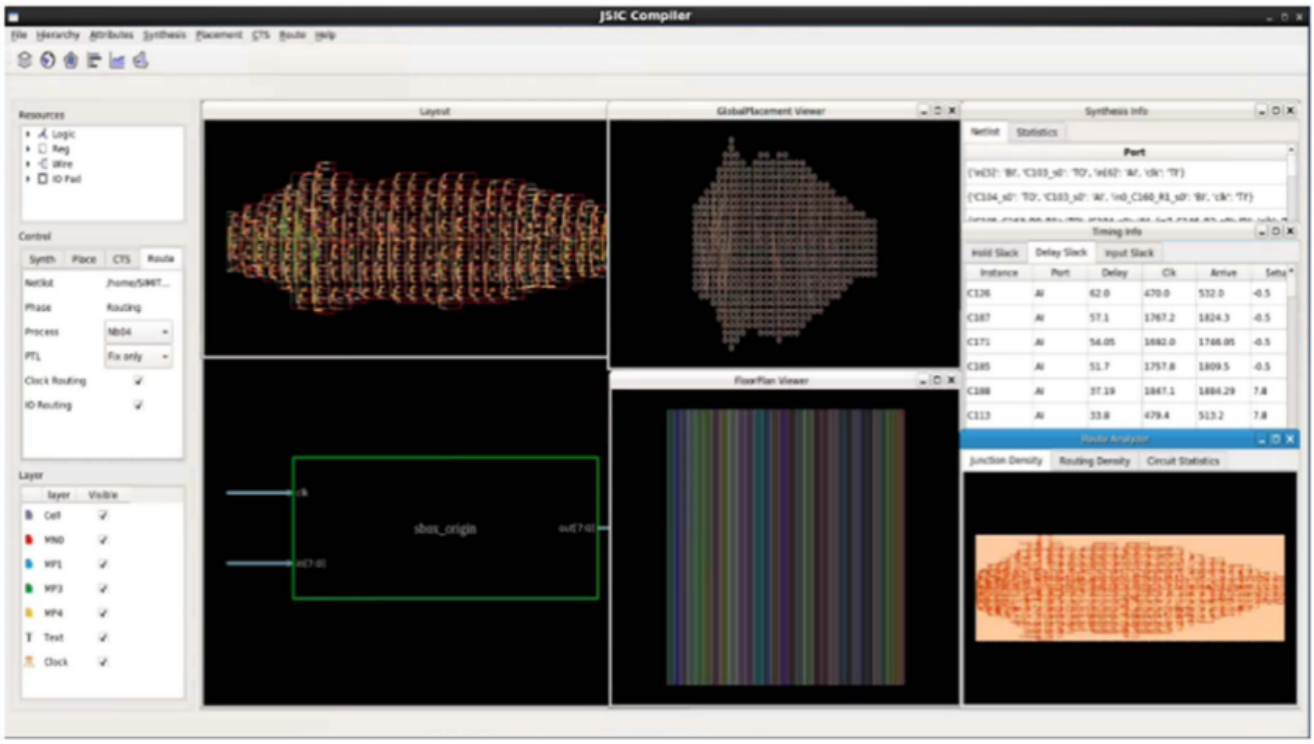

Fig. 1. Interface of JSICcomplier.

1. 具有映射的标准单元的网表[12],[13]将通过插入时钟端口、DFFs（用于路径平衡）和分频器树到SFQ网表来进行处理，这在放置之前完成。
2. 在出现误操作的情况下，将进行网表优化。将创建楼层计划，并进行全局放置。
3. 在上述所有步骤完成后，将生成时钟树。之后，路由过程将运行到具有详细的时序和分析报告的最终结果。 对于定制要求，为用户提供了一些选项，例如，在逻辑合成过程中时钟类型是可选的。此外，在时钟树合成中可以自由选择时钟的扇出。还提供了不同的布线方法。到目前为止，用户可以完成SFQ逻辑电路的自动设计过程。然而，逻辑合成和局部放置的详细优化仍在开发中。

- Netlist with mapped standard cells
    1. 网表 (Netlist)：它是电子系统的抽象描述，通常代表电路的逻辑或物理描述。在逻辑描述中，网表可能包括逻辑门、触发器等的连接。在物理描述中，网表会显示实际的电子组件，如晶体管和电阻，以及它们的连接。
    2. 标准单元 (Standard Cells)：这些是预先设计和特性化的数字逻辑块，如逻辑门、触发器和复用器。它们具有已知的物理尺寸和已知的逻辑功能，并用于半导体设计中。标准单元库提供了一组这些预定义的逻辑块，用于在更高层次的设计中使用
    3. 带映射的 (Mapped)：这意味着已经将高级逻辑设计（可能是RTL级别的描述）转换成一个使用标准单元库中的特定单元的具体描述。这个过程也叫做逻辑合成。 当我们说“带映射的标准单元的网表”时，我们指的是一个电路描述，该描述使用特定的标准单元来表示，并且已经经过逻辑合成的过程。这为后续的物理设计阶段，例如布局和布线，提供了详细的信息。

Case Study- AES S-Box Circuits

AES算法是一个先进的加密标准算法，于2001年由美国国家标准与技术研究院宣布 [14]。AES算法涉及四个单独的转换，其中SubBytes (0) 转换意味着每个状态字节都是非线性的，并且独立地使用一个称为S-box的替代表进行替代。

- 选择S-box实现电路作为测试电路的原因如下：

1. 由SFQ逻辑实施的加密算法加速器很少，仅在 [15] 和 [16] 中报告过。作为一个新的应用领域，设计问题也可以作为优化方法的灵感来源。

2. S-box电路设计中的规模和复杂性目前适用于SFQ逻辑电路。因此，它是探索AES算法在SFQ逻辑中的硬件实施并研究SFQ电路中的优化的第一步。

Local Optimization Method

主要思想

- 优化关注两个方面:
    1. 是否可以通过逻辑综合减少流水线阶段, 从而减少路径平衡的需要
    2. 是否可以直接解决DFF本身的问题 文中提出了一种方法, 旨在减少时钟节点并保持路径平衡功能, 以保持电路的吞吐量; IP可以替代DFF堆栈并保持电路的逻辑级别, 时钟节点将显著减少

同步器IP设计

- 使用同步器IP方法替代每个阶段的DFFs, 这种策略有助于减少电路中的时钟节点数并确保每个阶段的逻辑级别

测试组设计

- 每个同步器单元的成本比常见的DFF高
- 为了确定此方法是否有效, 设置了几个测试组来定量计算它们的消耗
- 使用同步器IP生成时钟树与使用多个时钟扇出点相似. 考虑了时钟树设计的因素, 如导线长度, 时钟扇出点数量和DFF的数量
- 什么是扇出? 扇出点Fan-out是指数字逻辑中一个逻辑门输出驱动的输入负载数量. 简单说, 扇出描述了一个特定输出被多少其他逻辑输入所使用. 例如，如果一个逻辑门的输出连接到五个其他逻辑门的输入，则该逻辑门的扇出为5。

  在数字电路设计中，每个门都有最大扇出规格，这意味着它能够驱动的其他输入的数量是有限的。超过这个扇出限制可能导致电路的性能下降，如速度变慢、信号失真等问题

  例如： 一个与门的输出连接到两个或门和三个非门的输入，则与门的扇出是5。

Results analysis

关怀与同步器IPs和扇出点优化的结果

多个同步器IPs

- 观察到使用同步器IP可以减少时钟树资源的消耗
- 虽然某些测试组的减少率超过了20%, 但使用Dsyn_2相比于期望的结果有所下降, 达到16.10%的JJ减少
- 当DFF单元的数量增加, 而且超过管道级别的一半以上时, 节点的数量会少于理论值
- 当使用更多的同步器单元时，如Dsyn_3和Dsyn_4，结果会更优。

多重时钟扇出

- 已证明多重时钟扇出可以减少电路成本。
- 文章考虑了结合IP进行多重扇出的效果，结果显示混合控制组可以减少资源消耗

通用电路的结果

- 这部分描述了优化方法在不同类型和大小的电路上的适用性。
- 实验结果显示了在几个电路上时钟树的JJ消耗。
- 描述了与其他电路、如ISCAS 89， c17, c432和c499等，的对比。

分析

- 当电路中的阶段少于三个时，使用Dsyn_2仍然有优化效果。
- 根据电路的不同，选择不同的IP可以获得最佳的优化效果。
- 使用Dsyn系列可以减少时钟树的节点数量和功耗。
- 在使用Dsyn优化的大多数电路中，替换的DFFs可能不在关键路径上。
- 通过放置优化和时钟树同步算法，Dsyn系列可能会导致局部区域的拥堵。
- 什么是管道级别? Pipeline stage特指流水线设计中的一个处理阶段 流水线设计是一种将一个复杂的操作分解为多个简单的子操作, 每个子操作在连续的时钟周期内执行.

例如，考虑一个简单的处理器流水线，它可以被分为五个阶段：

1. 取指（Instruction Fetch）
2. 指令译码（Instruction Decode）
3. 执行（Execute）
4. 访问存储器（Memory Access）
5. 写回（Write Back） 这五个阶段合在一起组成了处理器的完整流水线，每个阶段代表了一个“管道级别”或“管道阶段”。