

doi:10.19665/j.issn1001-2400.2023.03.014

RELIC-GNN:一种高效的状态寄存器识别算法

董 勳¹, 高一鸣¹, 潘伟涛¹, 邱智亮¹,
杨建磊², 邱志雄³, 郑 凌⁴

- (1. 西安电子科技大学 空天地一体化综合业务网全国重点实验室, 陕西 西安 710071;
2. 北京航空航天大学 计算机学院, 北京 100191;
3. 西南交通大学 信息科学与技术学院, 四川 成都 611756;
4. 西安邮电大学 通信与信息工程学院, 陕西 西安 710121)

摘要: 随着集成电路(IC)设计水平化、制造全球化的发展,由第三方厂商生产的大量硬件集成电路被应用于芯片设计中,这引起了人们对芯片中被插入设计后门/硬件木马的担忧。逆向工程可以恢复出集成电路芯片的设计网表,设计人员通过提取高层描述并分析关键逻辑可以判断设计功能是否被篡改。然而,逆向网表的可读性差,其数据路径和控制逻辑混杂在一起,难以快速、准确地抽象出高层描述。文中将该问题等价定义为网表路径结构分类问题,并提出一种基于图神经网络的高效状态寄存器识别算法。首先对网表预处理,消除工艺库的差异并降低建模复杂度;其次将网表建模为有向图,并提取其中每个寄存器的路径结构;然后将路径结构输入到构建好的图神经网络模型中,为每个寄存器生成相应的特征;最后对嵌入的特征进行聚类,将寄存器分为状态寄存器和控制寄存器。实验结果证明,该算法可以在百万门级网表上正确运行,其平均识别准确率达到约88.37%,相较于现有算法,在识别精度、运行速度、可迁移性等方面均有提升。

关键词: 逆向工程;寄存器分类;控制逻辑提取;图神经网络

中图分类号: TN406 **文献标识码:** A **文章编号:** 1001-2400(2023)03-0142-09

RELIC-GNN: an efficient state register identification algorithm

DONG Meng¹, GAO Yiming¹, PAN Weitao¹, QIU Zhiliang¹,
YANG Jianlei², DI Zhixiong³, ZHENG Ling⁴

- (1. State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China;
2. School of Computer Science and Engineering, Beihang University, Beijing 100191, China;
3. School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China;
4. School of Communications and Information Engineering, Xi'an University of
Posts and Telecommunications, Xi'an 710121, China)

收稿日期:2022-08-22

网络出版时间:2023-05-10

基金项目:国家自然科学基金(62102314);陕西省自然科学基金基础研究计划(2021JQ-708);陕西省教育厅专项科研计划(20JK0923);榆林市科技计划(YF-2020-183)

作者简介:董 勳(1995—),男,西安电子科技大学博士研究生,E-mail:mdong@stu.xidian.edu.cn

高一鸣(1998—),男,西安电子科技大学硕士研究生,E-mail:gyming@stu.xidian.edu.cn

邱智亮(1965—),男,教授,E-mail:zqiu@mail.xidian.edu.cn

杨建磊(1987—),男,副教授,E-mail:jianlei@buaa.edu.cn

邱志雄(1984—),男,副教授,E-mail:dizhixiong2@126.com

郑 凌(1989—),男,讲师,E-mail:lingzheng@xupt.edu.cn

通信作者:潘伟涛(1981—),男,副教授,E-mail:wtpan@mail.xidian.edu.cn

网络出版地址:https://kns.cnki.net/kcms/detail/61.1076.TN.20230509.1435.006.html

Abstract : With the horizontalization of integrated circuit (IC) design and globalization of manufacturing, a large number of hardware ICs produced by third-party vendors are used in the chip design, which raises concerns about design backdoors/hardware Trojan horses being inserted into chips. Reverse engineering can recover the design netlist of IC chips, and designers can determine whether the design functions have been tampered with by extracting high-level descriptions and analyzing the key logic. However, the poor readability of the reverse netlist with its data paths and control logic mixed makes it difficult to abstract the high-level descriptions quickly and accurately. In this paper, the problem is equivalently defined as the classification problem of the netlist path structure, and an efficient state register identification algorithm based on the graph neural network is proposed. First, pre-processing of the netlist is conducted to eliminate the differences of the process library and to reduce the modeling complexity. Second, the netlist is modeled as the directed graph and the path structure of each register is extracted. Then the graph neural network model is used to map corresponding features of each register with the path structure inputted. Finally, the features are clustered so as to classify the registers into status registers and control registers. Experimental results prove that the algorithm can run correctly on a million-gate netlist with the average recognition accuracy reaching 88.37%, which is improved in recognition accuracy, operation speed and migratability compared with the existing algorithms.

Key Words : reverse engineering; register classification; control logic extraction; graph neural networks

1 引言

近年来,随着全球化的发展,第三方知识产权核和电子设计自动化工具、商用现成集成电路等被广泛应用于集成电路(Integrated Circuit, IC)设计中。为了进一步降低芯片开发成本和缩短营销周期,越来越多的集成电路公司在芯片的设计、制造、测试等方面采用全球外包。因此在芯片设计中,第三方厂商可以轻易地接触到集成电路产品,从而增加了插入硬件木马、窃取知识产权等的安全威胁^[1]。逆向工程是通过对集成电路芯片进行解析,恢复其电路结构,进而分析芯片的原始设计功能。虽然逆向工程常被非法用于盗版场景,但文中利用其成果为芯片的设计和制造提供可靠性保障,这对集成电路产业的发展有益。

集成电路的逆向工程通常包含网表提取和网表分析两个阶段^[2]。在网表提取阶段,得益于先进的成像技术,逆向工程工具通常可以较轻易地从制造好的集成电路芯片中提取到完整网表。对于文中的研究来说,设计者可以直接从制造商处拿到可测性设计(Design For Testability, DFT)之后的网表,避免通过成像技术获取网表的复杂过程。在网表分析阶段,主要目的是从可读性差的网表中获取高级信息,便于设计者分析芯片的电路结构与功能。网表通常由数据路径块和控制逻辑两部分组成,数据路径块常用于实现通用功能,主要由基本功能部件如加法器、乘法器、移位器等组成;而控制逻辑是为特定功能设计的,通常由有限状态机(Finite State Machine, FSM)进行建模。高效准确地识别网表中的控制逻辑,对划分电路结构、推导设计功能有很大帮助^[3-5]。

然而,逆向工程提取的网表通常是展平的。网表中的数据路径和控制逻辑完全混合在一起,给网表分析带来了许多困难。现已存在一些用于识别网表中状态寄存器的算法^[6-10],能够较准确地识别出小型网表中的状态寄存器。逆向工程逻辑识别与分类(Reverse Engineering Logic Identification and Classification, RELIC)^[6]是首次根据寄存器扇入结构的相似性,对信号进行分类的工作。其基本思想是通过混合使用动态编程技术和高级图形算法,根据扇入结构的相似性和输入门的类型对每两个寄存器计算 $[0, 1]$ 范围内的匹配相似性分数,再使用简单的分类器来识别寄存器的类型。快速逆向工程逻辑识别与分类(fastRELIC)^[7]是作为对 RELIC 性能增强而提出的,其改进了相似性分数的计算方法,并在分类时引入聚类思想,有效提高了识别与分类的效率。基于字级结构的逆向工程(WordRev)^[10]算法将网表划分为 k 个可行切割的等价类,并使每个类的所有成员都能计算相同的布尔函数。该算法通过尝试寻找类中成员的连接来进行分类,数据路径的各个部分通过前向传播的方式进行连接,即从一个字的寄存器输出搜索到附近的信号,从而找到路径

<http://journal.xidian.edu.cn/xdxh>

中的下一个字,进而找到网表中的数据路径。上述这些状态寄存器识别算法在小型网表上取得了不错的识别结果,然而,对于百万门级别的大型设计网表,这些算法无法在可接受的时间内完成寄存器的识别。

针对上述问题,笔者提出了一种基于图神经网络的高效的状态寄存器识别算法(Reverse Engineering Logic Identification and Classification using Graph Neural Networks,RELIC-GNN)。该算法将寄存器路径结构映射为有向图特征,以图分类的方式对任意无标签网表中的寄存器进行快速分类。在 RELIC-GNN 中,首先将网表建模为有向图,并提取网表中每个寄存器的路径结构。然后将这些路径结构输入到构建好的图神经网络模型中,为每个寄存器生成相应的特征。最后,根据电路中每个寄存器的嵌入特征,通过聚类的方式将寄存器分为状态寄存器和数据寄存器两部分。文中的主要贡献如下:

(1) 提出了一种逆向工程分析算法 RELIC-GNN,将网表建模为有向图结构,将寄存器分类问题转换为图分类问题,并使用图神经网络(GNN)算法识别网表中的状态寄存器。

(2) 提出的算法可应用于多种器件工艺库,具有良好的可迁移性。

2 基于图神经网络的逻辑识别与分类算法

文中提出的逻辑识别与分类算法可以根据路径结构的不同,区分网表中的数据逻辑与控制逻辑。图 1 说明了文中提出逻辑识别与分类算法 RELIC-GNN 的框架。首先使用一组基准电路训练用于分类的图神经网络模型,然后利用训练好的 GNN 模型对新网表进行推理,最后通过分类算法,输出可能的控制逻辑和数据逻辑。训练的第一步是从输入的门级网表中提取出每个寄存器所在的路径结构,将其转换为有向图,得到一个有向图集合,并提取有向图中每个节点对应的特征。第二步,将集合中的有向图分别传递给 GNN 框架,首先通过“门嵌入”的方式为每个节点生成嵌入特征,然后通过解码步骤,恢复嵌入后的节点特征,并生成一个新的图。第三步,通过最小化节点恢复特征和初始特征之间的差值,及两图之间的差异进行训练。在 GNN 执行推理步骤后,通过分类算法识别网表中所有的控制寄存器与数据寄存器。

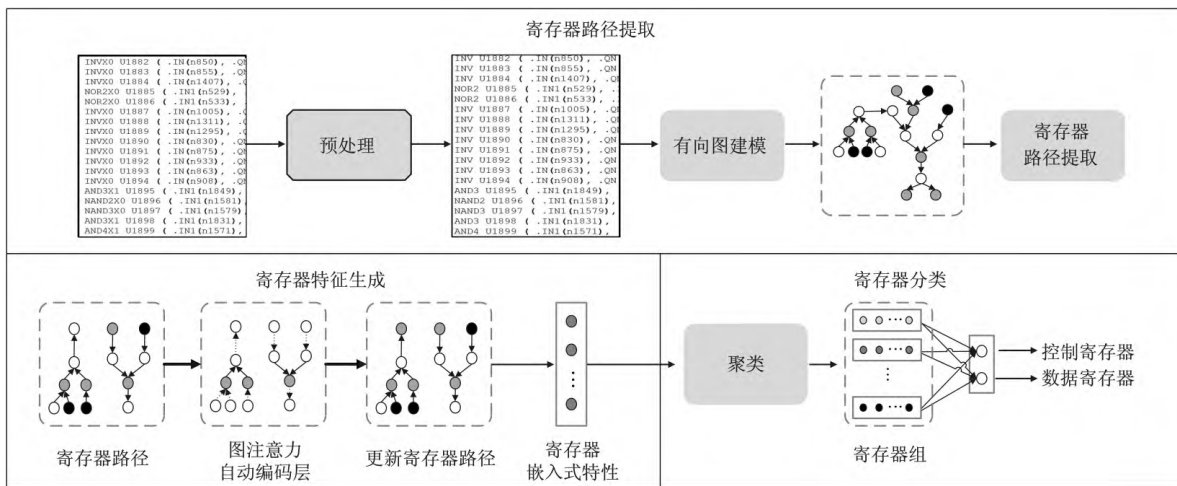


图 1 基于图神经网络的逻辑识别与分类算法流程

2.1 路径结构提取

在集成电路设计中,具有相同逻辑功能的单元,由于器件的时序、面积、功率等方面的差异,在同一个库中会有多个描述。网表分析中只关注了单元的逻辑功能,为降低路径结构提取与建模的复杂度,笔者对现有单元库进行分析与学习,建立了一个与技术无关的单元模型库;再对输入门级网表进行预处理,将所有单元进行替换,得到独立于技术库的网表。

在预处理之后,将门级网表转换为有向图 $G=(V,E)$,其中 V 是门级网表中所有节点的集合, E 是逻辑门之间的数据依赖集合;再提取所有寄存器的路径结构,从而对电路逻辑进行划分。在电路设计中,控制逻辑

辑主要负责产生或发送关键信号以及控制相关的数据通路,因此为了避免高延迟,控制逻辑的路径结构相对较短。对于多数设计网表,一个最大深度为 6 的路径结构可以包含所有控制逻辑。文中采用深度优先搜索的方式进行路径查找,当达到最大深度 6 或搜索到其他寄存器时则停止。

2.2 特征选择与提取

根据路径结构和逻辑单元的功能属性、有向图中节点统计信息以及节点关系,为每一个节点产生特征向量。节点特征向量主要包含以下信息:

- (1) 节点类型:使用独热码表示节点的类型,其维度是单元模型库中单元的种类;
- (2) 节点入度:节点传入邻居的数量,即节点对应单元的扇入数量;
- (3) 节点出度:节点传出邻居的数量,即节点对应单元的扇出数量;
- (4) 节点邻居的个数:该特征维度和节点类型维度相同,统计节点两跳内存在各类节点的个数。

2.3 模型建立与测试

图神经网络是针对图领域的深度学习模型,其核心思想是邻域聚合和特征嵌入。在图建立时每个节点或者边均会被分配一个特征向量,称为初始特征嵌入。邻域聚合主要是采用消息传递的方式来实现,通过更新函数对目标节点及其邻居节点的嵌入特征进行聚合与更新。而特征嵌入是指在训练过程中,将邻域聚合后的节点特征从高维向量映射成低维向量,从而使节点嵌入后的特征更好地表示节点本身及其结构。图神经网络通过节点之间的消息传递机制来捕获整个图的结构,并将经过几次聚合后的最终嵌入特征用来执行如节点分类、图分类等特定的任务。不同的图神经网络架构,如图卷积网络(Graph Convolutional Network, GCN)、图注意力网络(Graph Attention networks, GAT)等,均使用不同的领域聚合和节点嵌入函数。

在逻辑识别与分类问题中,人工参与的识别状态寄存器泛化能力差且准确度低。因此,笔者首先采用无监督学习的方式为每个寄存器生成相应的嵌入特征。如图 2 所示,无监督学习包括编码和解码两部分:编码通过堆叠多个 GAT 层,为每个节点生成相应的嵌入特征;解码通过相同数量的 GAT 层,将嵌入后的特征恢复为初始特征。通过最小化编码前与解码后节点特征之间的差异以及图结构重建的损失,可以获得最优的节点嵌入表征。最后,由于文中所解决的任务属于图分类问题,通过 READOUT 函数为编码后的图生成特征。

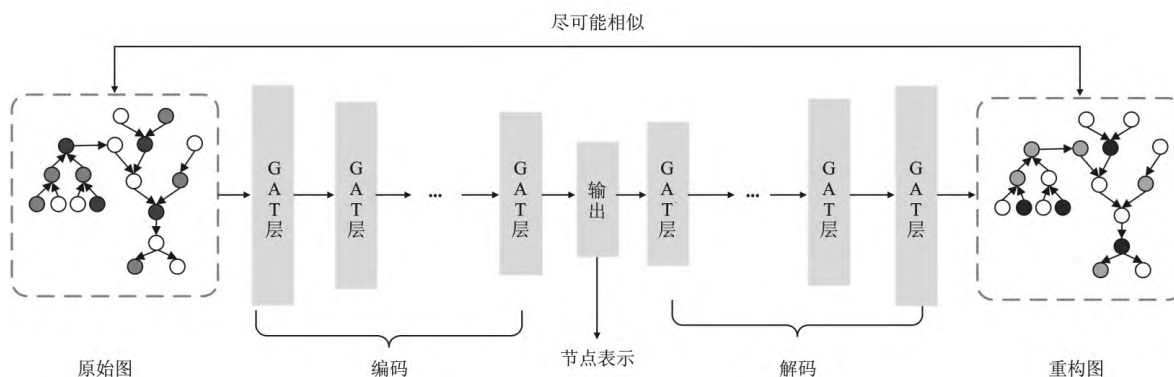


图 2 无监督学习框架

2.3.1 编码部分

为提升模型学习和表示能力,编码部分叠加多个 GAT 层以获取节点的嵌入特征,再通过 READOUT 函数对嵌入特征进行整合以作为图的编码结果。传统 GCN 利用拉普拉斯矩阵进行计算,无法处理动态图,并且为邻居节点分配不同权重时的实现代价较大,这些都限制了 GCN 算法的效果。文中使用文献[11]提出的 GAT 模型,在节点聚合时引入注意力机制。该方法虽然增加了算法计算量,但有利于获取邻居节点的特征信息。对于图中节点 i 来说,逐个计算其邻居节点 j 对节点 i 的重要性,如式(1)所示:

$$e_{ij}^{(k)} = a(\mathbf{W}^{(k)} \mathbf{h}_i^{(k)}, \mathbf{W}^{(k)} \mathbf{h}_j^{(k)}) \quad (1)$$

首先通过各节点之间的共享权重 \mathbf{W} 将节点 i 和其邻居节点 j 的特征进行维度变化,并将更新后的特征进行拼接;然后通过一个单层前馈神经网络 $\alpha(\cdot)$ 将拼接后的特征映射为一个实数;为了便于比较不同节点之

间的系数,将获取后的重要性通过 softmax 函数进行归一化,得到节点 i 和其邻居节点 j 之间的注意力系数, softmax 函数如式(2)所示:

$$\alpha_{ij}^{(k)} = \text{softmax}_j(e_{ij}^{(k)}) = \frac{\exp(e_{ij}^{(k)})}{\sum_{p \in \mathcal{N}_i} \exp(e_{ip}^{(k)})} \quad (2)$$

最后,根据获取到的注意力系数,对节点 i 所有的邻居节点 j 的特征进行如式(3)所示的加权求和,得到节点 i 的最新特征:

$$\mathbf{h}_i^{(k+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j^{(k)}\right) \quad (3)$$

其中, σ 是激活函数,通常可以为 RELU、Sigmoid 等。此外,为了增加算法的健壮性,在编码过程中可以选择使用多头注意力机制。

编码模型的输入为节 2.1 提取的 6 级路径结构进行集合,路径深度过大会导致 GAT 训练效果平滑,降低模型准度。为了让节点感知其所有相邻节点,同时保证模型的精度,编码部分使用 3 个堆叠的 GAT 层,非线性激活函数均使用 RELU 函数。在第 1 层中,为了增加节点的表示能力,节点特征被映射至 64 维;在第 3 层中,为了降低后续计算的复杂性,线性映射节点特征均被压缩至 1 维。

2.3.2 解码部分

在无监督学习的场景下,只有通过嵌入后特征尽可能地恢复出原始特征,才能保证节点和图的表示被模型正确学习。因此,解码部分使用的解码器数量与编码部分的编码器数量一致,每个解码器都尝试通过节点及其邻居节点的特征反向执行编码过程以重建节点的表示。式(4)~式(6)说明了这一过程,其中 k 表示当前解码器层数(为了保证参数统一,解码器层数是递减的)。解码器初始输入为编码器最后编码的结果,且每层的输入为前一层解码器的解码结果:

$$\hat{e}_{ij}^{(k)} = a(\hat{\mathbf{W}}^{(k)} \hat{\mathbf{h}}_i^{(k)}, \hat{\mathbf{W}}^{(k)} \hat{\mathbf{h}}_j^{(k)}) \quad (4)$$

$$\hat{\alpha}_{ij}^{(k)} = \text{softmax}_j(\hat{e}_{ij}^{(k)}) = \frac{\exp(\hat{e}_{ij}^{(k)})}{\sum_{p \in \mathcal{N}_i} \exp(\hat{e}_{ip}^{(k)})} \quad (5)$$

$$\hat{\mathbf{h}}_i^{(k-1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \hat{\alpha}_{ij}^{(k)} \hat{\mathbf{W}}^{(k)} \hat{\mathbf{h}}_j^{(k)}\right) \quad (6)$$

2.3.3 损失函数

有向图相同意味着两个图的节点特征和边特征完全一致,但在文中使用的无监督学习框架中,有向图只有节点被赋予了初始特征,而边只体现了节点之间的数据流走向,并没有特征表示。此外,文中编码没有使用池化等技术,所有相邻节点均参与了特征嵌入的计算。因此,在构建重建损失时,只需将所有节点之间的差异相加,即

$$R_{\text{Loss}} = \sum_{i=1}^N \|\mathbf{v}_i - \hat{\mathbf{v}}_i\|_2 \quad (7)$$

2.4 寄存器分类

利用节 2.3 中训练好的模型,输入网表中每一个寄存器对应的路径结构均可以被映射为一个特征值。文中对所有寄存器的特征值进行比较和聚类,从而识别数据寄存器与控制寄存器。

具体实现方式如下:将所有的路径结构全部标记为候选路径结构(Candidate Path Structure, CPS),并将其放入到候选组中(Candidate Group, CG)。首先,随机从 CG 中选择一个路径结构作为起始路径结构(Start Path Structure, SPS),新建一个空集合 S_i 并将 SPS 放入其中。然后,计算 SPS 与 CG 中所有路径结构之间特征差异(Feature Difference Value, FDV)的绝对值,并将所有 FDV 小于阈值 T_1 的路径结构从 CG 移至集合 S_i 中,其余所有路径结构保留。迭代上述过程,直到 CG 中没有剩余的寄存器结构。对于同一组总线的数据寄存器,其路径结构相似但不一定完全相同,例如加法器输出数据最高位的逻辑与其他位不同。在分组过程中,为了尽可能区分数据和控制逻辑,需设置一个接近 0 的阈值 T_1 ,文中阈值 T_1 设置为 10^{-3} 。

<http://journal.xidian.edu.cn/xdxh>

数据逻辑通常其路径结构是相似的,而控制逻辑的路径结构一般有较大区别。因此,统计每个集合 S_i 内的路径结构数量,若小于阈值 T_2 ,则将该组内所有路径结构归类为控制逻辑;反之则归类为数据逻辑。在芯片设计中,数据逻辑的位宽通常是大于等于 4 bit,因此文中将 T_2 设置为 4。

2.5 算法复杂度分析

假设网表中的寄存器数量为 R 。在路径结构提取中,每个寄存器对应的节点信息均以字典的形式存储,查找给定节点的扇入节点的时间复杂度仅 $O(1)$ 。结合芯片的功耗、面积、性能等因素,节点扇入的数量通常不超过 4,对现有的工艺库如 SMIC 130、SMIC 28 的分析也证实了这一点。每个寄存器都需要搜寻 6 级内的扇入结构,由此可以得出乘法系数 R_{MF_1} 表示为

$$R_{MF_1} = \left(\sum_{i=1}^{6-1} 4^i \right) R \quad (8)$$

GAT 模型不需要进行特征分解、拉普拉斯运算等时间复杂度较高的矩阵运算,其节点特征嵌入的复杂度为 $O(NFF' + EF')$,其中 N 为有向图节点数量, F 为节点初始特征维度, F' 为嵌入后节点特征维度, E 为有向图边数量。每个寄存器对应的有向图结构均需参与一次模型运算,由此可以得出乘法系数 R_{MF_2} 表示为

$$R_{MF_2} = \left(\sum_{i=1}^{6-1} 4^i FF' + \sum_{i=1}^{6-1} 4^i F' \right) R \quad (9)$$

寄存器分类的时间复杂度取决于特征差异值的计算次数。在最好的情况下,所有寄存器路径结构相同,只产生一个路径结构组,需要比较的次数 R_{LB} 如式(10)所示;在最差的情况下,所有寄存器的路径结构无相似性,需要比较的次数 R_{UB} 如式(11)所示:

$$R_{LB} = R - 1 \quad (10)$$

$$R_{UB} = 0.5R(R - 1) \quad (11)$$

综上,RELIC-GNN 算法时间复杂度的下限 O 和上限 O 分别如式(12)和式(13)所示:

$$O = OR_{MF_1} + OR_{MF_2} + O(R - 1) = O(R) \quad (12)$$

$$\Omega = OR_{MF_1} + OR_{MF_2} + O(0.5R(R - 1)) = O(R^2) \quad (13)$$

3 实验结果与分析

针对提出的基于图神经网络的逻辑识别与分类算法的性能进行了评估,并将其与 fastRELIC 和采用图神经网络的寄存器识别(Register Identification using Graph Neural Networks, ReIGNN)^[9]两个基线算法进行比较。实验所用数据是通过 Synopsys Design Compiler 软件综合生成的 flatten 网表,使用的工艺库为 SMIC 28 nm。

3.1 数据集与评价标准

文中所选取的数据包括来自 ITC99 的标准电路^[12](★),来自 OpenCore^[13](†)的通用开源设计,来自 secworks 的电路设计^[14](◇)、来自 32 bit 的 RISC-V 处理器^[15](‡)中的部分代码以及一个椭圆曲线数字签名算法^[16](§)。训练模型时,通过手动分析 RTL 代码得到状态寄存器的数量,通过提取分析综合后的网表信息得到寄存器和逻辑门的数量。模型训练完成后则不再需要手动分析代码。

实验时采用了留一交叉验证法验证算法在不同设计上的可迁移性,即在所给的 19 个网表中选择 18 个进行训练,剩余 1 个网表作为测试集,对训练好的模型进行测试。实验使用真阳性(R_{TP})、真阴性(R_{TN})、假阳性(R_{FP})和假阴性(R_{FN})来计算算法的评价指标。文中采用的评价指标为召回率(P_{Recall})、精确率($P_{Precision}$)和准确率($P_{Accuracy}$),其计算公式如下:

$$P_{Recall} = R_{TP} / (R_{TP} + R_{FN}) \quad (14)$$

$$P_{Precision} = R_{TP} / (R_{TP} + R_{FP}) \quad (15)$$

$$P_{Accuracy} = (R_{TP} + R_{FP}) / (R_{TP} + R_{TN} + R_{FP} + R_{FN}) \quad (16)$$

<http://journal.xidian.edu.cn/xdxh>

3.2 实验结果与性能分析

由于实验环境对算法效率及实验结果的影响很大,笔者并没有直接使用 fastRELIC 和 ReIGNN 所展现的结果,而是根据文献中的描述,手动复现这两个基线算法。需要注意的是,fastRELIC 算法中需要对 3 个阈值进行配置,文中采用的配置为 $T_1 = 0.7, T_2 = 0.5, T_3 = 4$ 。

表 1 RELIC-GNN 和两个基线算法对比结果

%

网表	召回率			精确率			准确率		
	Fast- RELIC	ReF- GNN	RELIC- GNN	Fast- RELIC	ReF- GNN	RELIC- GNN	Fast- RELIC	ReF- GNN	RELIC- GNN
FSM†	100	100	100	80.00	100	100	64.29	71.43	71.43
b10★	75.00	100	100	25.00	44.44	50.00	36.84	60.87	61.90
b08★	100	100	100	13.33	25.00	22.22	34.78	65.22	60.87
b09★	100	100	100	40.00	28.57	40.00	83.33	76.67	83.33
gpio†	100	75.00	100	26.67	33.33	66.67	68.75	80.43	87.50
b04★	100	100	100	11.76	18.18	22.22	75.00	83.82	86.76
MEM†	100	100	100	30.77	30.77	44.44	82.89	82.89	88.16
b14★	100	100	100	8.33	4.35	3.85	94.44	89.35	87.96
spi_axi†	100	100	100	22.22	7.27	33.33	96.75	90.07	97.83
uart†	100	100	100	21.21	2.85	53.81	94.6	89.74	97.87
siphash◇	100	100	100	6.12	3.37	15.79	93.81	88.76	97.60
mem_control†	74.24	96.97	100	27.07	29.09	25.29	84.33	81.43	78.10
alto32†	100	66.67	100	4.23	2.99	4.92	88.66	89.26	90.26
sha1◇	100	100	100	2.70	1.20	13.33	95.59	89.06	99.02
gcm_aes†	100	100	100	4.17	5.75	29.41	85.99	89.72	98.02
lightweight†	100	100	100	1.74	1.14	3.33	91.31	90.05	95.46
aes◇	100	100	100	6.06	2.81	22.86	95.59	90.48	98.83
cr_div†	100	100	100	3.71	0.77	27.27	98.06	90.61	99.74
escda §			100			0.58			98.47
平均	97.18	95.20	100	18.62	18.99	30.49	81.37	83.30	88.37

表 1 是 RELIC-GNN 算法和两个基线算法在召回率、精确率和准确率方面的对比结果。可以看出,提出的 RELIC-GNN 算法在评价召回率、精确率和准确率上均优于两个基线算法。相比于 fastRELIC 算法和 ReIGNN 算法来说,RELIC-GNN 算法具有更加均衡的分类表现。此外,由于算法的时间复杂度更低,RELIC-GNN 算法对大规模网表(如网表 escda)有着更显著的性能提升,而 fastRELIC 算法和 ReIGNN 算法则均无法在合理的时间(即 24 小时)内得到分类结果。

表 2 从运行时间的角度对 RELIC-GNN 算法和 fastRELIC 算法进行了对比。可以看出,RELIC-GNN 算法在多数情况下实现了明显的加速比,这在网表 mem_control 和 gcm_aes 中表现得尤其明显。分析 RTL 代码发现,这些设计具备大位宽的数据总线,相比于 fastRELIC 算法,RELIC-GNN 算法对有着大规模数据路径电路的处理时间更短。

此外,为了进一步评估所提出方法的可移植性,使用 SMIC 130 nm 的工艺库对选取的基准电路重新综合生成新的网表,再将新网表分别送入已经训练好的 GNN 模型中进行预测,预测结果如表 3 所示。可以看出,大多数网表的预测结果与原工艺库下的预测结果差异约在 5% 以内,这说明提出的算法具有可迁移性。部分电路如 b10 的差异较大,这是由于 28 nm 工艺下的设计网表被插入了更多的缓冲器,改变了寄存器的路径结构,从而影响迁移精度。

<http://journal.xidian.edu.cn/xdx>

表2 运行时间对比结果 s

网表	Fast-RELIC	RELIC-GNN	加速比例
FSM‡	1.76	1.84	0.96
b10★	2.64	3.38	0.93
b08★	2.27	3.43	0.85
b09★	3.35	3.40	1.06
gpio‡	1.83	3.57	0.82
b04★	5.83	3.67	1.63
MEM‡	10.59	8.34	1.27
b14★	28.36	4.21	6.90
spi_axi‡	43.96	5.70	9.16
uart†	76.38	10.56	7.23
siphash◇	252.92	6.51	41.73
mem_control†	355.25	7.75	50.61
alto32†	232.65	20.48	11.36
sha1◇	118.17	9.39	13.87
gcm_aes†	458.52	11.02	48.42
lightweight†	436.63	18.23	23.96
aes◇	281.51	15.87	19.87
cr_div†	529.03	22.19	26.63

表3 对工艺库的迁移性评估 %

网表	准确率	
	130 nm library	28 nm library
FSM‡	63.47	71.43
b10★	47.06	61.90
b08★	66.67	60.87
b09★	89.69	83.33
gpio‡	95.45	87.50
b04★	83.69	86.76
MEM‡	80.45	88.16
b14★	93.46	87.96
spi_axi‡	97.45	97.83
uart†	95.33	97.87
siphash◇	96.07	97.60
mem_control†	75.39	78.10
alto32†	86.76	90.26
sha1◇	94.16	99.02
gcm_aes†	93.49	98.02
lightweight†	83.33	95.46
aes◇	95.25	98.83
cr_div†	99.66	99.74
escda §	94.78	98.47
平均	85.57	88.37

4 结束语

针对芯片生产和制造过程中可能会被不可信的第三方厂商插入恶意逻辑导致功能错误、信息泄露等问题,在逆向工程的基础上提出了一种使用图神经网络识别逆向网表中的控制逻辑和数据逻辑的 RELIC-GNN 算法。首先对输入的网表进行预处理,消除工艺库中功能相同单元的电气特性差异,并将预处理后的网表转化为非欧空间的有向图结构。其次,在对路径结构进行充分分析的基础上,介绍了所提出的 RELIC-GNN 方法中每个节点的特征类型及含义。最后,针对网表中数据寄存器与控制寄存器分类困难的问题,引入无监督学习的思想,提高算法识别精度和可迁移性。实验结果表明,提出的算法在识别准确率和运行时间上较原有算法有较大改善,且在不同工艺库上具有较强的迁移能力。


参考文献:

- [1] BHUNIA S, HSIAO M S, BANGA M, et al. Hardware Trojan Attacks: Threat Analysis and Countermeasures [J]. Proceedings of the IEEE, 2014, 102(8): 1229-1247.
- [2] ALBARTUS N, HOFFMANN M, TEMME S, et al. DANA Universal Dataflow Analysis for Gate-Level Netlist Reverse Engineering [J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 4: 309-336.
- [3] MCELVAIN K S. Methods and Apparatuses for Automatic Extraction of Finite State Machines; US06182268B2 [P]. 2001-1-30.
- [4] SHI Y, TING C W, GWEE B H, et al. A Highly Efficient Method for Extracting FSMs from Flattened Gate-Level Netlist [C]//Proceedings of 2010 IEEE international symposium on circuits and systems. Piscataway: IEEE, 2010: 2610-2613.
- [5] MEADE T, ZHANG S, JIN Y. Netlist Reverse Engineering for High-Level Functionality Reconstruction [C]//2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Piscataway: IEEE, 2016: 655-660.
- [6] MEADE T, JIN Y, TEHRANIPOOR M, et al. Gate-Level Netlist Reverse Engineering for Hardware Security: Control

<http://journal.xidian.edu.cn/xdxh>

- Logic Register Identification[C]//2016 IEEE International Symposium on Circuits and Systems (ISCAS). Piscataway: IEEE, 2016: 1334-1337.
- [7] BRUNNER M, BAEHR J, SIGL G. Improving on State Register Identification in Sequential Hardware Reverse Engineering[C]//2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). Piscataway: IEEE, 2019: 151-160.
- [8] MEADE T, SHAMSI K, LE T, et al. The Old Frontier of Reverse Engineering: Netlist Partitioning[J]. Journal of Hardware and Systems Security, 2018, 2(3): 201-213.
- [9] CHOWDHURY S D, YANG K, NUZZO P. ReIGNN: State Register Identification Using Graph Neural Networks for Circuit Reverse Engineering[C]//2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD). Piscataway: IEEE, 2021: 1-9.
- [10] LI W, GASCON A, SUBRAMANYAN P, et al. WordRev: Finding Word-Level Structures in a Sea of Bit-Level Gates[C]//2013 IEEE international symposium on hardware-oriented security and trust (HOST). Piscataway: IEEE, 2013: 67-74.
- [11] VELICKOVIC P, CUCURULL G, CASANOVA A, et al. Graph Attention Networks (2017)[J/OL]. [2017-10-30]. <https://arxiv.org/abs/1710.10903>.
- [12] CORNO F, REORDA M S, SQUILLERO G. RT-Level ITC99 Benchmarks and First ATPG Results[J]. IEEE Design & Test of computers, 2000, 17(3): 44-53.
- [13] OPENCORE. Opencore benchmarks (2022)[EB/OL]. [2022-06-19]. <https://opencores.org>.
- [14] STRÖMBERGSON J. Secworks (2022)[EB/OL]. [2022-06-19]. <https://github.com/secworks?tab=repositories>.
- [15] ONCHIPUIS. Onchipuis (2022)[EB/OL]. [2022-06-19]. <https://github.com/onchipuis>.
- [16] DONG M. The netlist of ecdsa circuit (2022)[EB/OL]. [2022-06-19]. <https://github.com/Melvin-Dong/ecdsa>.

(编辑: 牛姗姗)



欢迎订阅

西安电子科技大学学报

JOURNAL OF XIDIAN UNIVERSITY

投稿邮箱: <http://journal.xidian.edu.cn/xdxb/CN/1001-2400/home.shtml>

<http://journal.xidian.edu.cn/xdxb>