MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Revisiting Graph Neural Networks for Netlist Reverse Engineering

Simon Klix

MPI-SP, Bochum, Germany

DISCLAIMER

This is a talk about work that is still in progress!

Netlist Reverse Engineering

Motivation:

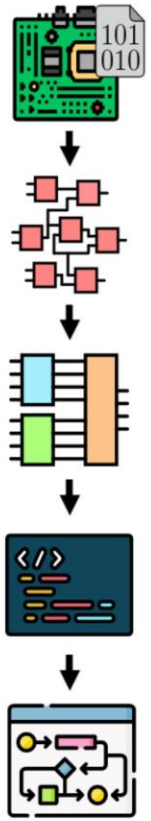
- You are here.

Problem:

- Loss of information during synthesis
- Size and Complexity

Solution:

- So far mostly hand crafted algorithms
- What about machine learning?



Input: Device / Bitstream

Step 1: Netlist Recovery
↳ **Output:** Gate-Level Netlist (*sea-of-gates*)

Step 2: Word-Level Reconstruction
↳ **Output:** Word-Level Netlist (*sea-of-modules*)

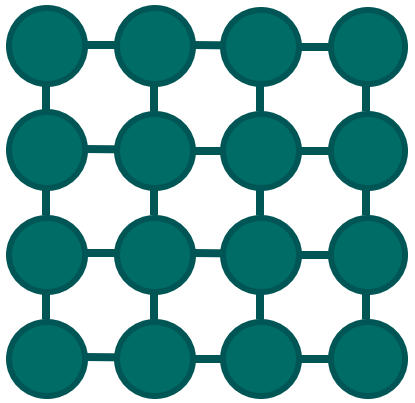
Step 3: Algorithmic Recovery
↳ **Output:** Algorithmic Description

Step 4: High-Level Sensemaking
↳ **Output:** High-Level Understanding

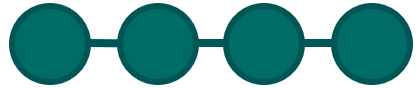
Result: Potential IP-Theft

Machine Learning for Netlist Reverse Engineering

- Machine Learning achieved impressive results in recent years for:
  - Images
  - Text



Image

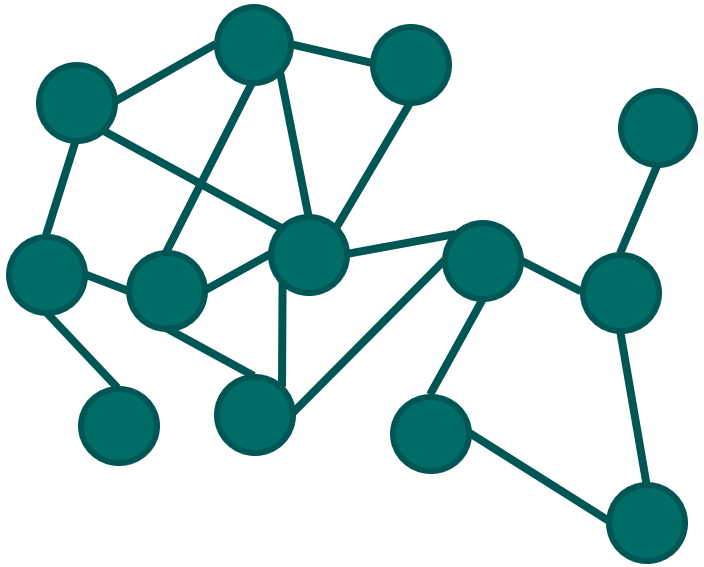


Text

Machine Learning for Netlist Reverse Engineering

- Machine Learning achieved impressive results in recent year for:
  - Images
  - Text

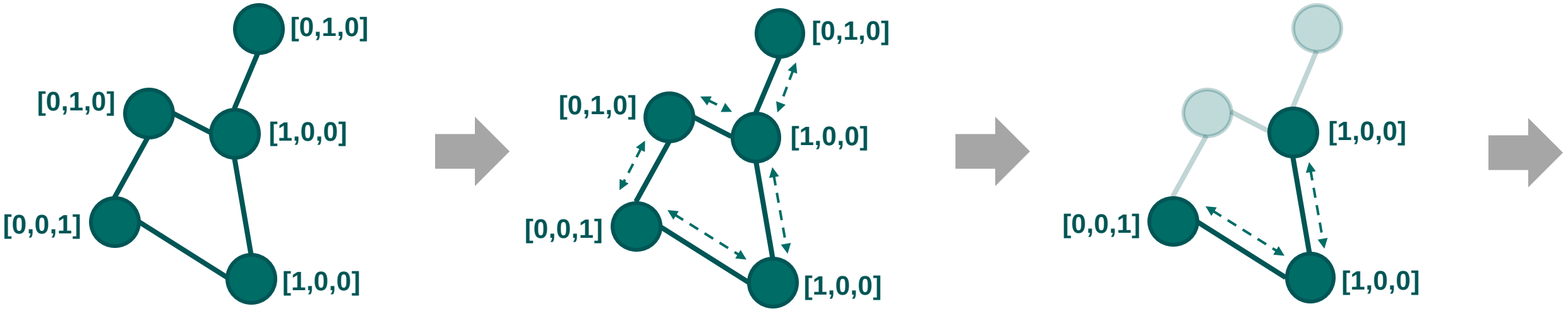
- Netlists are very **unstructured/heterogeneous**



Solution:

- A special type of machine learning model called Graph Neural Networks (GNNs)

Graph Neural Networks?

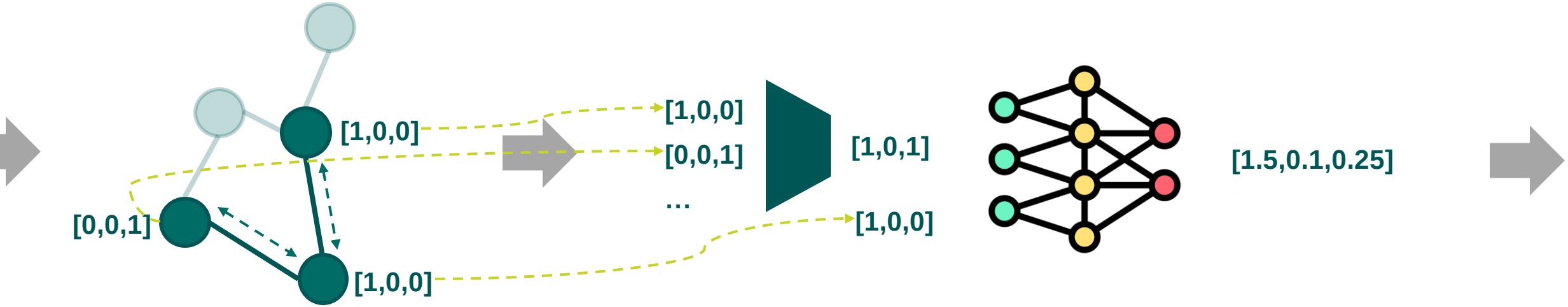


Inital Features

Message Passing

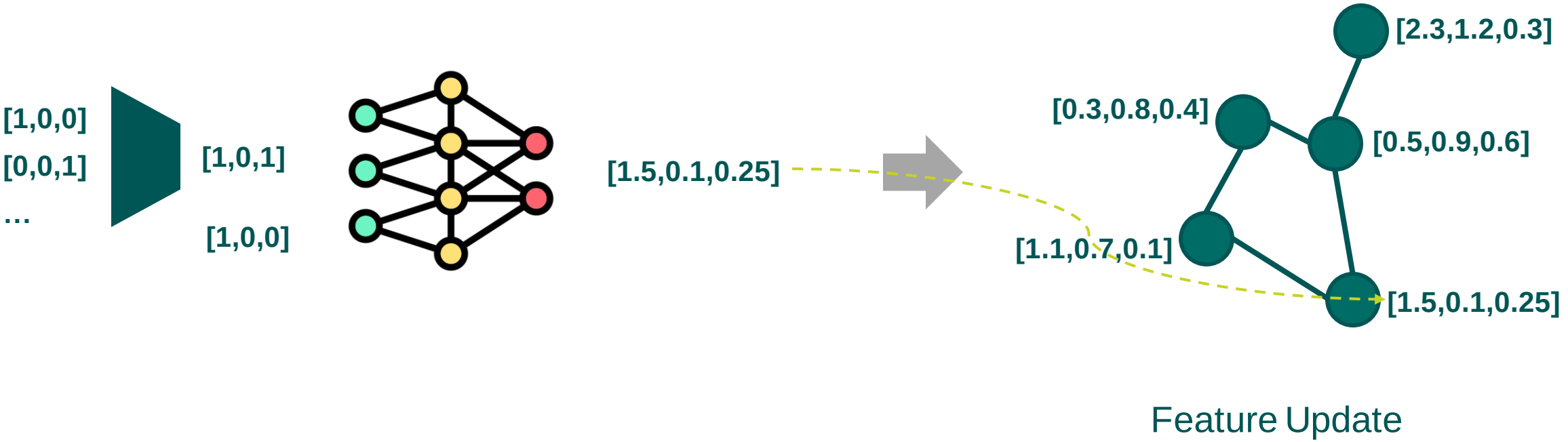
Message Aggregation

Graph Neural Networks?



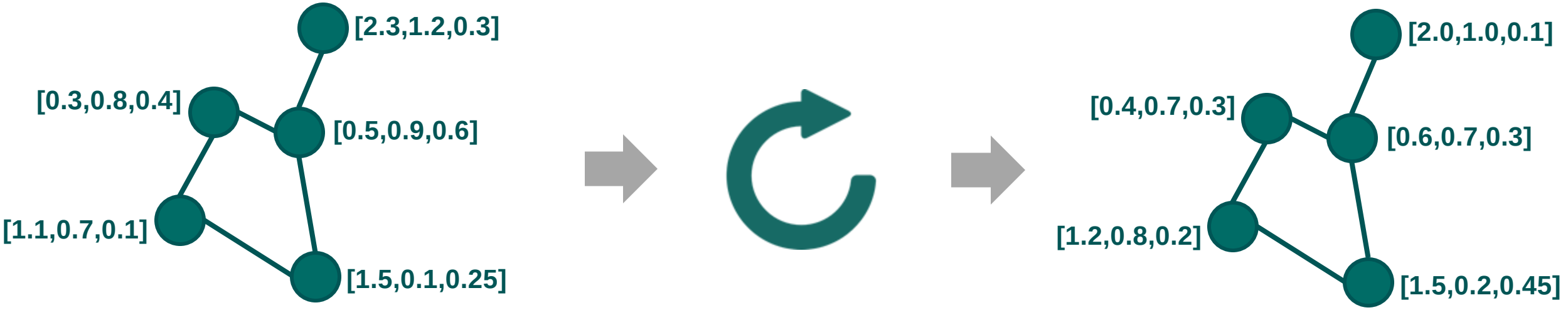
Message Aggregation

Graph Neural Networks?



Feature Update

Graph Neural Networks?

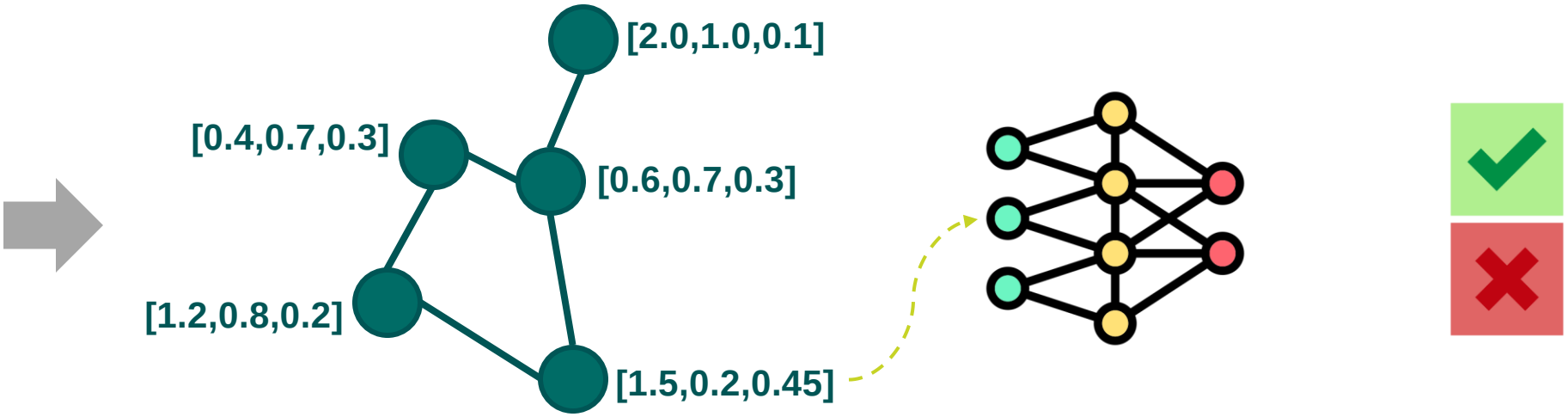


Feature Update

Repeat

Final Query

Graph Neural Networks?



Final Query

Why Revisting?

Identified Research Opportunities:

- Application to real-world netlists
- Workflow to all technologies / gate libraries
- Cover more aspects of netlist RE
- Easier to reproduce

ReIGNN: State Register Identification Using Graph Neural Networks for Circuit Reverse Engineering

Subhajit Dutta Chowdhury, Kaixin Yang, Pierluigi Nuzzo
Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA
{duttacho, kaixinya, nuzzo}@usc.edu

IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 41, NO. 8, AUGUST 2022 2435

GNN-RE: Graph Neural Networks for Reverse Engineering of Gate-Level Netlists

Lilas Alrahis, *Member, IEEE,* Abhrajit Sengupta, *Member, IEEE,* Johann Knechtel, *Member, IEEE,* Satwik Patnaik, *Member, IEEE,* Hani Saleh, *Senior Member, IEEE,* Baker Mohammad, *Senior Member, IEEE,* Mahmoud Al-Qutayri, *Senior Member, IEEE,* and Ozgur Sinanoglu, *Senior Member, IEEE*

***Abstract*—This work introduces a generic, machine learning (ML)-based platform for functional reverse engineering (RE) of circuits. Our proposed platform GNN-RE leverages the notion of graph neural networks (GNNs) to: 1) represent and analyze flattened/unstructured gate-level netlists; 2) automatically identify the boundaries between the modules or subcircuits implemented in such netlists; and 3) classify the subcircuits based on their functionalities. For GNNs in general, each graph node is tailored to learn about its own features and its neighboring nodes, which is a powerful approach for the detection of any kind of subgraphs of interest. For GNN-RE, in particular, each node represents a gate and is initialized with a feature vector that reflects on the functional and structural properties of its neighboring gates. GNN-RE also learns the global structure of the circuit, which facilitates identifying the boundaries between subcircuits in a flattened netlist. Initially, to provide high-quality data for training of GNN-RE, we deploy a comprehensive dataset of foundational designs/components with differing functionalities, implementation styles, bit widths, and interconnections. GNN-RE is then tested on the unseen shares of this custom dataset, as well as the EPFL benchmarks, the ISCAS-85 benchmarks, and the 74X series benchmarks. GNN-RE achieves an average accuracy of 98.82% in terms of mapping individual gates to modules, all without any manual intervention or postprocessing. We also release our code and source data.**

***Index Terms*—Gate-level netlist, graph neural networks (GNNs), hardware security, machine learning (ML), reverse engineering (RE).**

Manuscript received 27 October 2020; revised 16 June 2021; accepted 30 August 2021. Date of publication 7 September 2021; date of current version 19 July 2022. This work was supported in part by the High-Performance Computing resources at Khalifa University and in part by the Center for Cyber Security (CCS) at New York University Abu Dhabi (NYUAD). This article was recommended by Associate Editor L. C. Wang. *(Corresponding author: Lilas Alrahis.)*

Lilas Alrahis was with the System on Chip Center, Khalifa University, Abu Dhabi, UAE. She is now with the Division of Engineering, New York University Abu Dhabi, Abu Dhabi, UAE (e-mail: lma387@nyu.edu).

Abhrajit Sengupta was with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA. He is now with Qualcomm, San Diego, CA, USA (e-mail: as9397@nyu.edu).

Johann Knechtel and Ozgur Sinanoglu are with the Division of Engineering, New York University Abu Dhabi, Abu Dhabi, UAE (e-mail: johann@nyu.edu; orgunsin@nyu.edu).

Satwik Patnaik was with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA. He is now with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: satwik.patnaik@tamu.edu).

Hani Saleh, Baker Mohammad, and Mahmoud Al-Qutayri are with the System on Chip Center, Khalifa University, Abu Dhabi, UAE.

Digital Object Identifier 10.1109/TCAD.2021.3110807

I. Introduction

With the continuous escalation in complexity and costs for integrated circuit (IC) fabrication, the semiconductor manufacturing process has become globalized, involving numerous entities across the globe. Such a globalized supply chain leads to a plethora of security-related concerns, such as illegal overproduction of ICs, piracy of design intellectual property (IP), or implantation of malicious hardware Trojans (HTs)[1] [4], [5], any of which can cause financial loss to design companies or IP owners and introduce potential risks to end users. For example, the financial losses for US companies due to infringement of design IP were estimated as U.S. $225–600 billion in 2017 [6]. To this end, researchers have proposed various countermeasures (see [7] for a detailed review).

Reverse engineering (RE) of an IC is a process that aims to obtain the IC/IP design, technology, or functionality by analyzing the chip layer by layer [8]. RE of ICs can be leveraged to detect IP infringement, verify IP implementation, detect HTs, etc. A typical RE flow is illustrated in Fig. 1. The first stage is to obtain the gate-level netlist from the physical chip (steps ❶-❸ in Fig. 1). See [9] and [10] for more details. Note that the netlist can also be reverse engineered directly from the layout data (GDSII file), as illustrated by the black dashed lines in Fig. 1 [11]. The second stage is functional RE, i.e., ascertaining the functionality of the chip/GDSII (steps ❹ and ❼ in Fig. 1). Further details are discussed in [12]–[15].

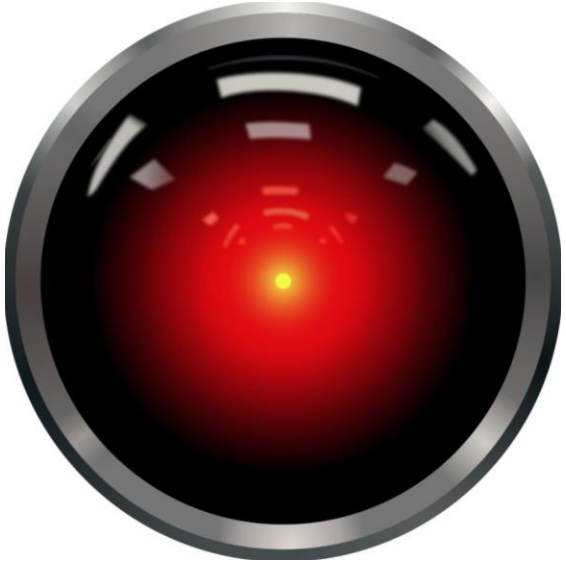
In this work, we focus on functional RE (blue-dashed box in Fig. 1). Prior art typically applies the following work flow: first, a set of candidate subcircuits is extracted, e.g., by partitioning the netlist, and then each subcircuit is labeled, e.g., by performing exhaustive formal verification against components from a golden library [16]–[19]. These works have the following limitations: 1) extracting all the relevant candidate subcircuits and checking each candidate by formal verification is a time-consuming procedure. The performance and accuracy of such an approach depend heavily on the constructed golden library and 2) such techniques cannot identify any variants of design components in the golden library [20].

[1]HTs are malicious circuit modifications that can be introduced during fabrication, through adversarial third-party IP cores, untrusted or "hacked" design tools, or by malign employees during chip fabrication [2]. The general objectives of HTs are to modify the chip's functionality or steal critical data/assets processed on the chip [3].

GNNs for Netlist Reverse Engineering

Integration into existing open-source infrastructure allows:

- Reusing general purpose parser and existing gate libraries
- Easy pre- and postprocessing
- Possibility to add different features and labels
- Collaboration and reproducability



GNNs for Netlist Reverse Engineering

Research Question I:
What features can we use to annotate our gates with and which ones are the most effective?

- Features used so far include:
  - Gate types
  - Connection to global IO
  - Graph centrality metrics

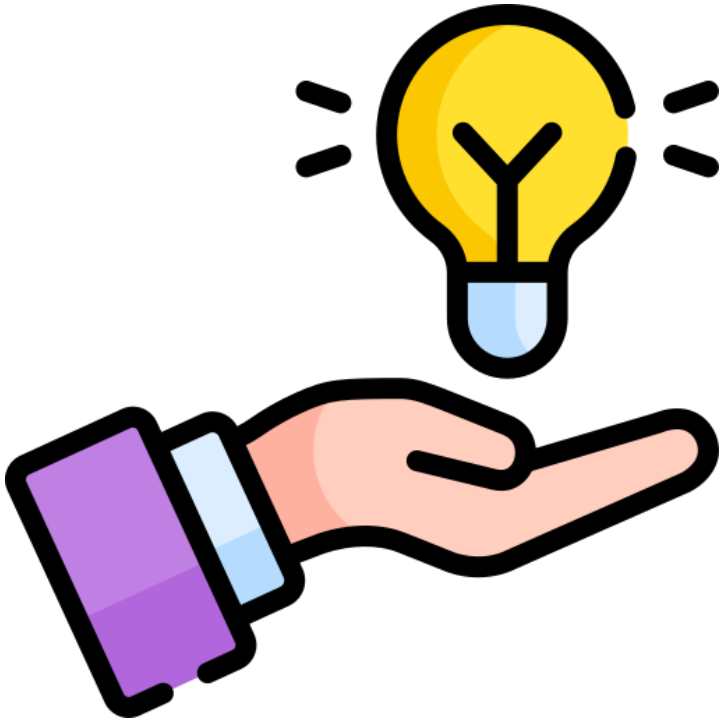
- Features we want to experiment with:
  - Distance to other elements in the netlist
  - Physical Placement on the chip

GNNs for Netlist Reverse Engineering

Research Question II:
What netlist reverse engineering problems are we able to solve with GNNs and to what extent?

- Problems considered in academia include:
  - Control logic identification
  - Arithmetic logic identification

- Problems we want to try and solve:
  - Register identification
  - Bitorder reconsturction



GNNs for Netlist Reverse Engineering

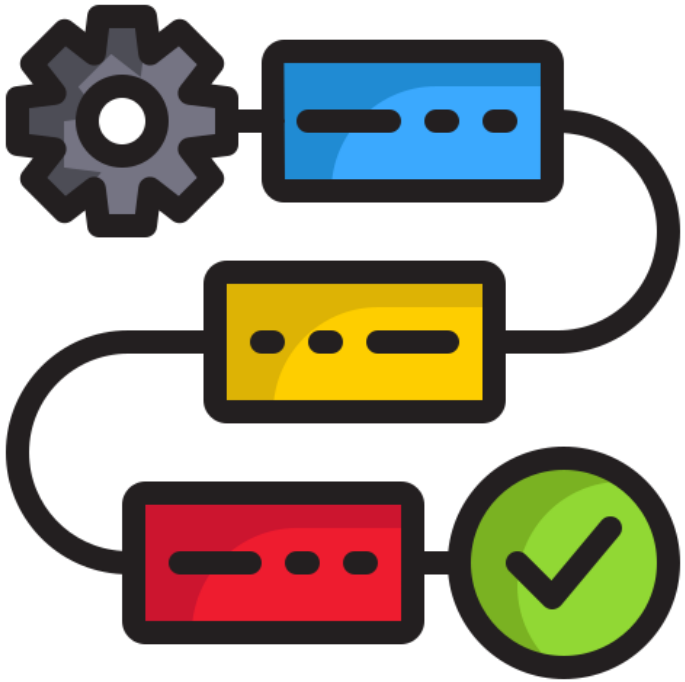
Research Question III:
How can we train robust models with minimal manual labelling and limited access to netlist designs.

- We try to leverage synthetic data
  - Amount of hardware primitives is finite
  - Framework that generates complex netlists with characteristics similar to real netlists
  - When being in control during design we can insert labels into the RTL

GNNs for Netlist Reverse Engineering

Envisioned workflow:

- Extract netlist from target
- Create gate library for target
- Create (resynthesize) dataset for target gate library
- Retrain models for specific target



How is it Going?

Experiments running at the moment showing promising results:

- Reproducing and generalizing existing work
- Register grouping (95% accuracy on synthetic and real data)

Experiments where we are still unsure about feasibility:

- Bitorder reconstruction (currently around 70% accuracy)

How is it Going?

Learnings so far:

- Deeper message propagation for netlist reverse engineering can lead to performance increase
- More expressive models indicate higher quality results
- Currently no performance discrepancy between ASIC and FPGA

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Any Questions ?