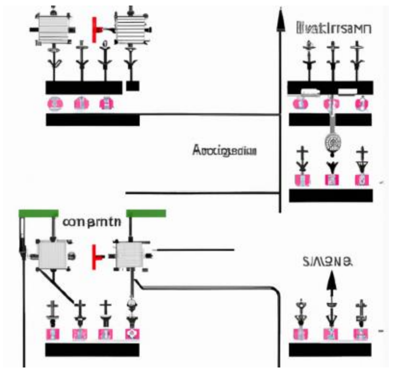
Automatic synthetic benchmark generator for hardware security



Background: Research in computer science and engineering requires data for evaluation. These data should be consistent, organized and non-biased to serve a base line for evaluation of tools, algorithms and general research. In particular, hardware security and reverse engineering needs benchmarks for research and evaluation. Recently, with the introduction of machine learning to these fields, an additional important use of benchmarks has emerged, which is training set. For example, detection of familiar structures within a logical circuit can be done using a Graph-Neural Network (GNN), which is trained with examples of such structures.
However, number of logical circuits openly available is limited and therefore cannot serve a sound base for training. The industry needs a way to obtain benchmarks in large amounts. A possible solution is creating synthetic benchmarks based on some reference authentic examples and functionally ‘similar’ to them.

Project Description: In this project, the students will build a tool that generates synthetic benchmarks. The tool will get at its input a reference benchmark and provide a set of synthetic benchmarks ‘similar’ to the reference. Since functional similarity is more a perceptional than a scientific term, it will be achieved in an indirect way. For that, the original circuit will be first converted to an AND-Inverter graph. This graph will be structurally modified using graph algorithms under constraints. The modified graph will be converted back to the original gate library. Logic synthesis tools, such as Synopsys Design Compiler can be used to perform the conversion, while the graph manipulation can be done using Python with the relevant graph algorithm packages.
Possible continuation of the project is evaluation of the quality of the tool by using the generated benchmarks to train a real GNN to detect subcircuits.
In the course of the project, the students will acquire knowledge logic synthesis, circuit analysis and graph algorithms.

Prerequisites: Logic Design, Algorithms, Machine Learning (recommended), Lab 1.

Supervisor: Leonid Azriel (*leonida@technion.ac.il*)