Hardware Trojan Detection

Barak Binyamin

Contents

- Methodology
- Results
- Analysis

Methodology

Identify

- Pseudorandom Input/Output Comparison

Characterize

- Infected output analysis (which bits are effected), what percentage of outputs are effected (indicating sensitivity of trojan -> either looser combinational logic or sensitive to more inputs)
- Infected input analysis (try to locate patterns in bit position values inputs)
- Sequential input analysis (try to locate patterns in bit position values using N=1 previous outputs as additional inputs)

Results

```
------------------- Trojan Report... -------------------

Trojan Detected


Given:
        Input: 32 bits
        Output: 32 bits

Analysis:
        It appears that the trojan effects the following bits of the output
                (0..N-1) (Msb to Lsb): [28]

        14 out of 1000, 1.40% of the outputs were effected

        The combinational input trigger bits are likely to include [13, 2, 29, 25, 6]

        If sequential, output[index-1] input trigger bits are likely to include [29, 25, 6, 13, 2, 1, 0]

------------------- End of Trojan Report... ----------------
```

Analysis

The detection methodology proved to be successful in identifying all trojan bitstreams and characterizing some key aspects of the trojans provided

The trojan detection and characterization methodology focused more on combinational trojans as it did not does not repeat input values intentionally. It also searched more heavily for signs of combinational triggers

Some of the detection methods were not fine tuned enough to consistently identify combinational trigger bits, the methodology was not fully automated and not design focused, as it did not take the netlist into consideration, for example to analyze rare nodes

Github

```
git clone https://github.com/BarakBinyamin/Trojan-Detection.git && cd Trojan-Detection
make
```

```
FPGA TROJAN DETECTION

Finding hardware trojans in FPGA bitsreams...
Made by Rocky https://linkedin.com/in/barak-binyamin-664a211a1
usage: make <option>
        s1     : Collect golden samples using psudorandom input
        t1     : Run simple tests comparing psudorandom input re
```