ECE407 - Computer Aided Design for VLSI

Assignment Report - Part A

Test Development using KNN, GNN

Student 1: Lefkios Mavroudi
UCY ID: UC1064650

Student 2: Alexandros Vryonides
UCY ID: UC1066645

Contents

1 Introduction

The objective of Part A is to construct the structural representation of digital circuits using ISCAS'85/89 benchmark files. We develop a parser for the `.bench` format, build the corresponding directed acyclic graph (DAG), perform topological traversal, and compute all primary input (PI) to primary output (PO) paths. This graph representation will later be used for machine learning tasks such as feature extraction, KNN classification, and GNN modeling.

2 Methodology

2.1 Parsing ISCAS Benchmark Files

To construct the structural graph of the circuit, we implemented a custom Python parser for ISCAS'85/89 `.bench` files. The parser processes the benchmark line-by-line and identifies three fundamental elements of the circuit:

- primary inputs (lines of the form `INPUT(x)`)
- primary outputs (lines of the form `OUTPUT(y)`)
- internal gates and their signal dependencies (e.g. `10 = NAND(1, 3)`)

Each `INPUT(k)` or `OUTPUT(k)` line is parsed by extracting the signal identifier inside the parentheses. These are stored in dedicated sets of primary inputs (PI) and primary outputs (PO).

For gate definitions, the parser handles lines of the general form:

```
x = GATE(a, b, ...)
```

The left-hand side (`x`) represents the output signal produced by the gate, while the right-hand side lists all the input signals that feed into it. For every input signal `u` feeding the gate, the parser records a directed edge

$$u \to x$$

indicating that the value on node x depends functionally on the value of u. All nodes and edges are inserted into Python sets, producing a clean representation of the circuit in graph form.

Because ISCAS benchmark circuits contain no cycles by construction, the resulting graph is a directed acyclic graph (DAG). This structural DAG is then used for all subsequent stages of the assignment: topological traversal, path counting, and later machine learning feature extraction.

2.2 Graph Construction: G1 Structural Graph

The resulting graph is the G1 structural graph, where:

- Nodes represent signals/wires
- Edges represent functional dependencies (input → gate output)

This graph supports:

- Topological traversal
- PI–PO path counting

2.3 Topological Order Traversal

We use a queue-based topological traversal algorithm as presented in the *Lecture 4 – Modelling* slides. Let $G(V, E)$ be the circuit graph, PI the set of nodes with no predecessors (primary inputs), and $S(v)$ the set of immediate successors of node v.

Algorithm 1 Topological_Traversal(G)

```
1: Q: queue;   v, p: node
2: Init_Queue(Q)
3: for every node v ∈ V do
4:    if v ∈ PI then
5:       Enqueue(Q, v); mark v as visited
6:    else
7:       mark v as not-visited
8:    end if
9: end for
10: while Empty_Queue(Q) == False do
11:    p ← Dequeue(Q)
12:    for every node v ∈ S(p) do
13:       if v not yet visited and v can be visited then
14:          Enqueue(Q, v); mark v as visited
15:       end if
16:    end for
17:    process p
18: end while
```

2.4 Path Counting Algorithm

Using the lecture's formula, we compute the number of PI→node paths.

For each node v:

$$p(v) = \begin{cases} 1, & v \in PI \\ \sum_{u \in pred(v)} p(u), & \text{otherwise} \end{cases}$$

Total number of PI→PO paths is:

$$\text{Total Paths} = \sum_{v \in PO} p(v)$$

This matches the algorithm described in the course slides.

3 Results

3.1 Verification on Benchmark c17

We tested our implementation using the ISCAS benchmark `c17.bench`. The following results were obtained:

- Correct directed acyclic graph construction
- Valid topological ordering
- Expected path counts

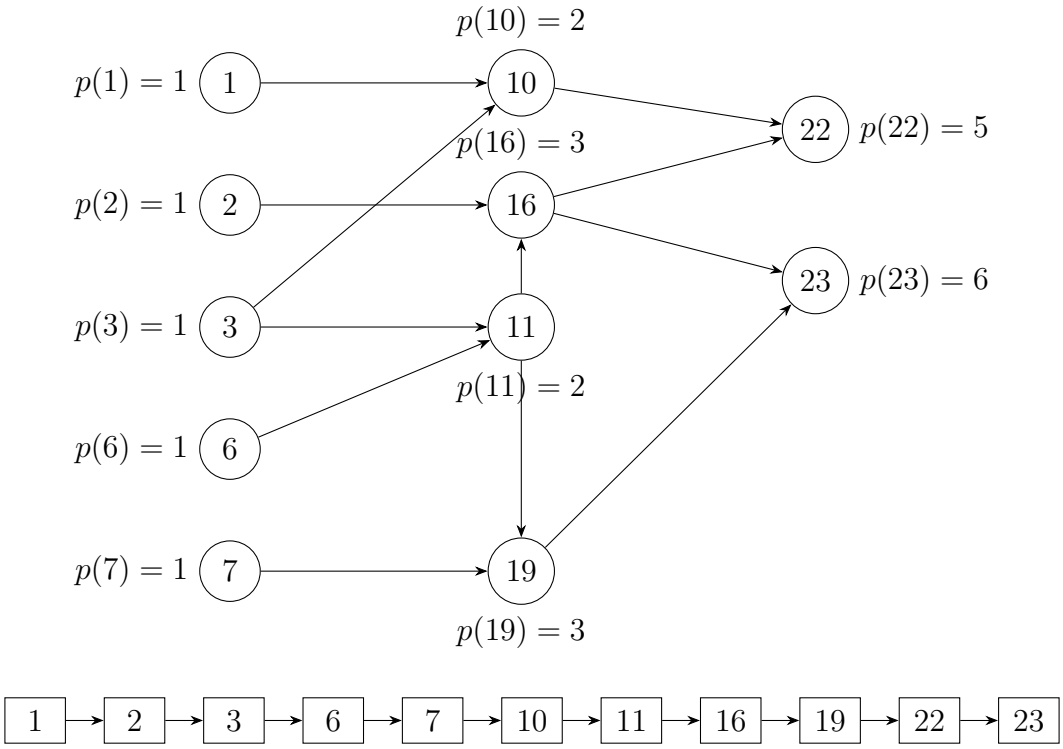

Figure 1: G1 structural graph of the c17 circuit with path counts $p(v)$ and one valid topological order queue.

Path counting results:

$$p(22) = 5, \qquad p(23) = 6$$

Total number of PI→PO paths:

$$\text{Total} = 11$$

These results confirm that the parser, graph construction, and traversal functions behave correctly.

4 Conclusion

In this part of the assignment, we successfully implemented:

- Parsing of ISCAS'85/89 benchmark circuits
- Construction of the G1 structural DAG
- Topological traversal
- Path counting

These results provide the necessary foundation for the upcoming machine learning components of the project, including KNN and GNN modeling.