Gate level circuit to linked list conversion

Overview

- Benchmarking
- Circuit benchmark formats
- Input parsing
- Recommended data structures
- Circuit/Graph path counting algorithm

Benchmarking

- Used to evaluate the performance of the developed CAD tool.
- Allows for better sharing and fair comparison of research results.
- Encourages healthy competition.
- Development of "*good*" benchmarks is an important research topic.

Circuit Benchmarks

- Some of the circuit benchmarks that have been used extensively to evaluate VLSI test related research:
  - ISCAS’85: Combinational circuit profiles
  - ISCAS’89: Sequential circuits; we will use the full-scanned versions
  - ITC’99: The most recent and most challenging!

ISCAS: International Symposium of Circuits And Systems

ITC: International Test Conference

ISCAS'85/ISCAS'89/ITC'99 Bench Format

Circuit c17 from the ISCAS'85 suite

Header

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )
```

Primary Inputs

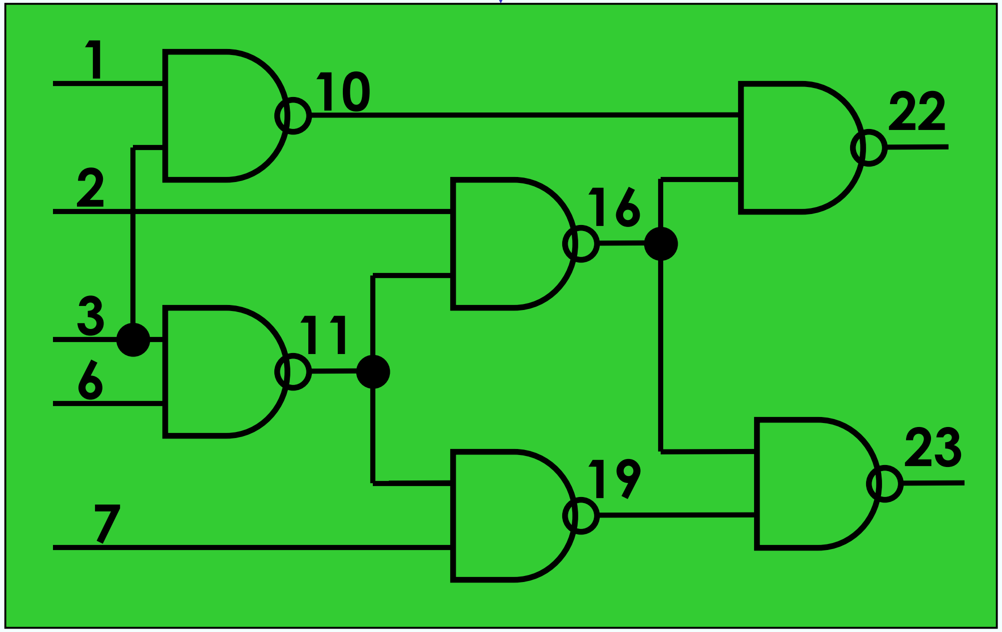
```
INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)
```

Primary Outputs

```
OUTPUT(22)
OUTPUT(23)
```

Gates

```
10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```



ISCAS'85/ISCAS'89/ITC'99 Bench Format

- —— keyword
- —— variable

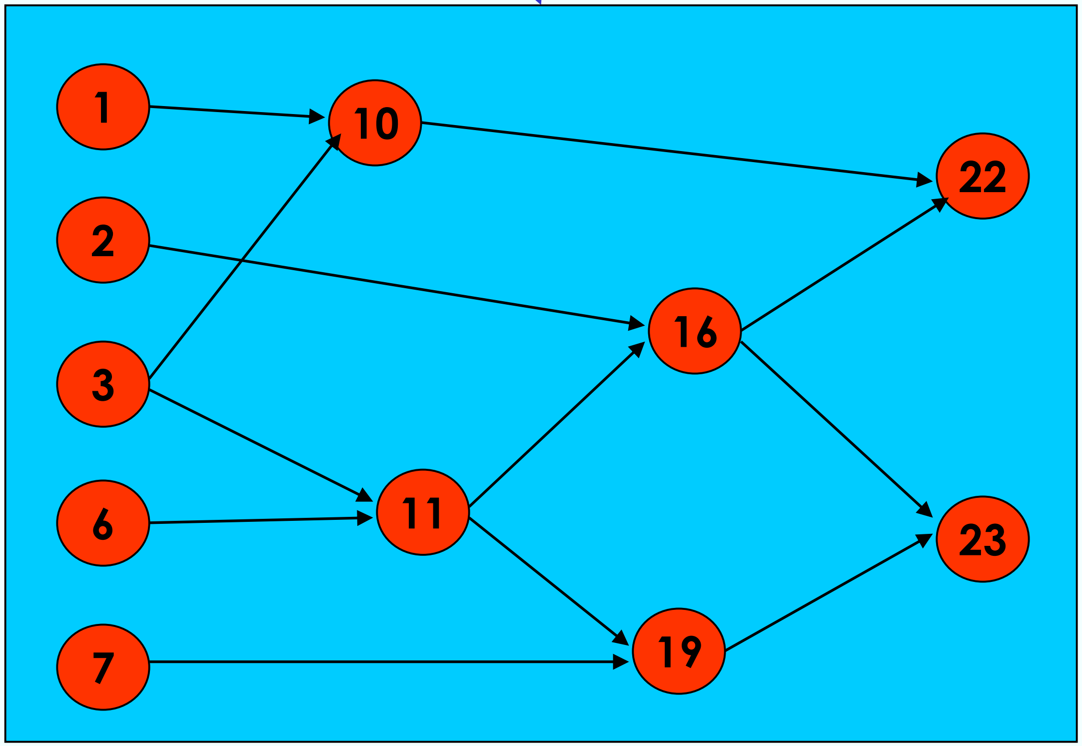
- # - comment line
- INPUT(x) - primary input line with name x
- OUTPUT(x) - primary output line with name x
- x = GTYPE(IN1, IN2, …, INk) - gate of type GTYPE with output x and k inputs, IN1…INk
- Valid gate type GTYPE

  AND, NAND, OR, NOR, XOR, XNOR, NOT, BUFF

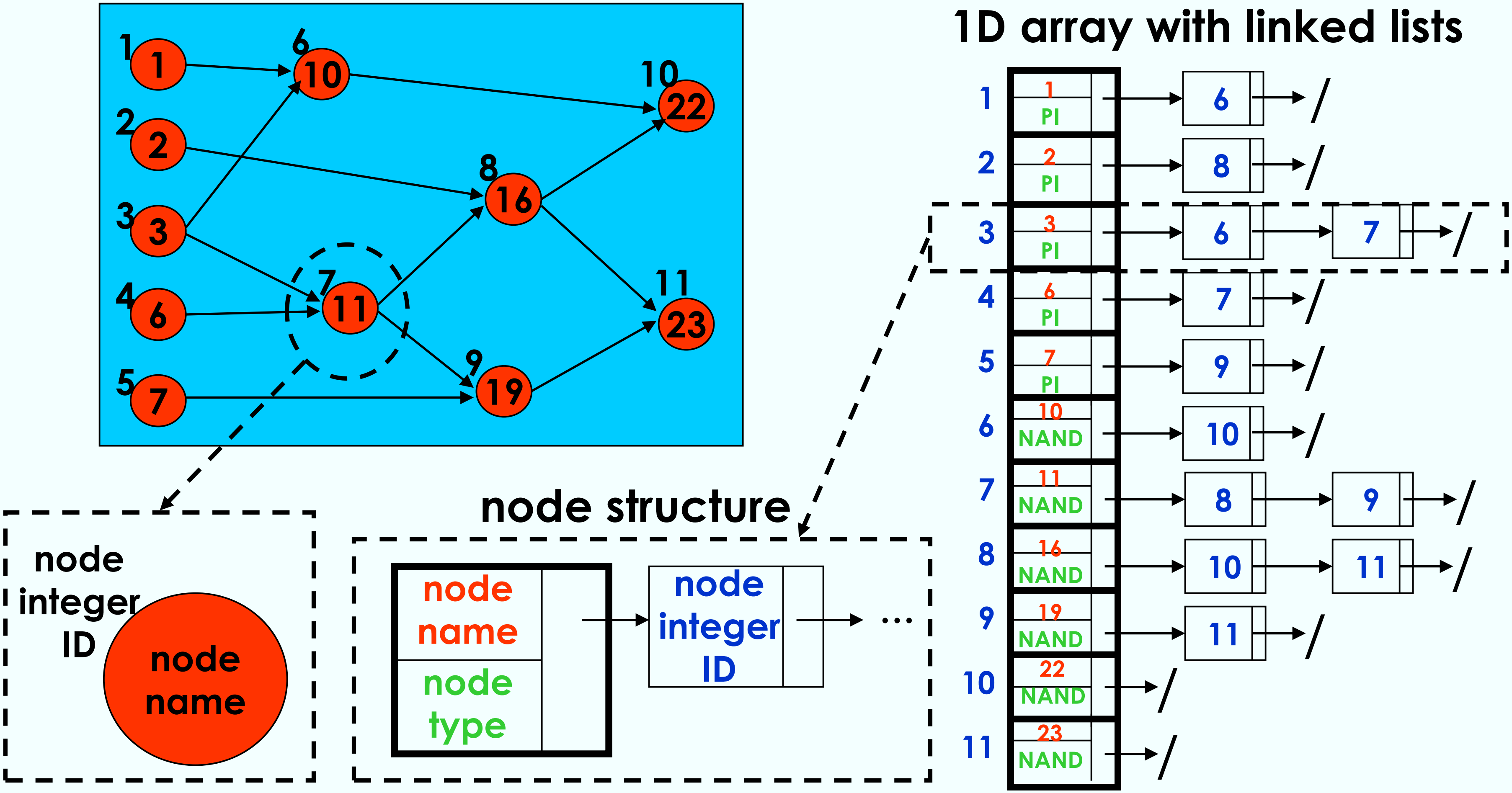
Parse bench circuit and store as graph

```
# c17
# 5 inputs
# 2 outputs
# 0 inverter
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)

OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

Circuit c17 from the ISCAS'85 set

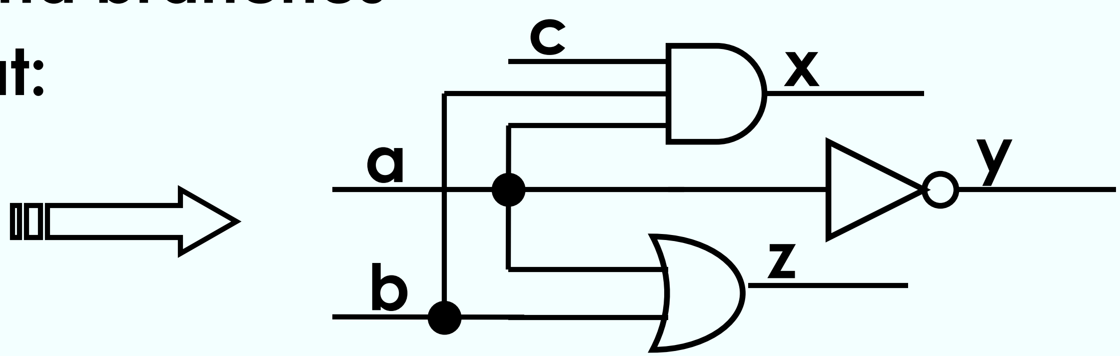


Recommended data structures (example for circuit c17)

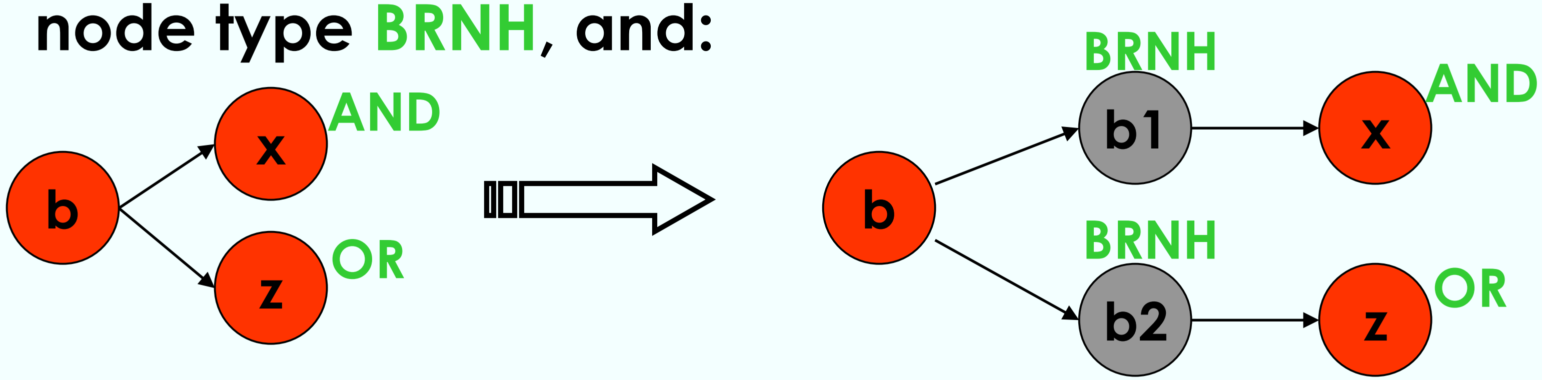


Adding BRANCH nodes to the graph

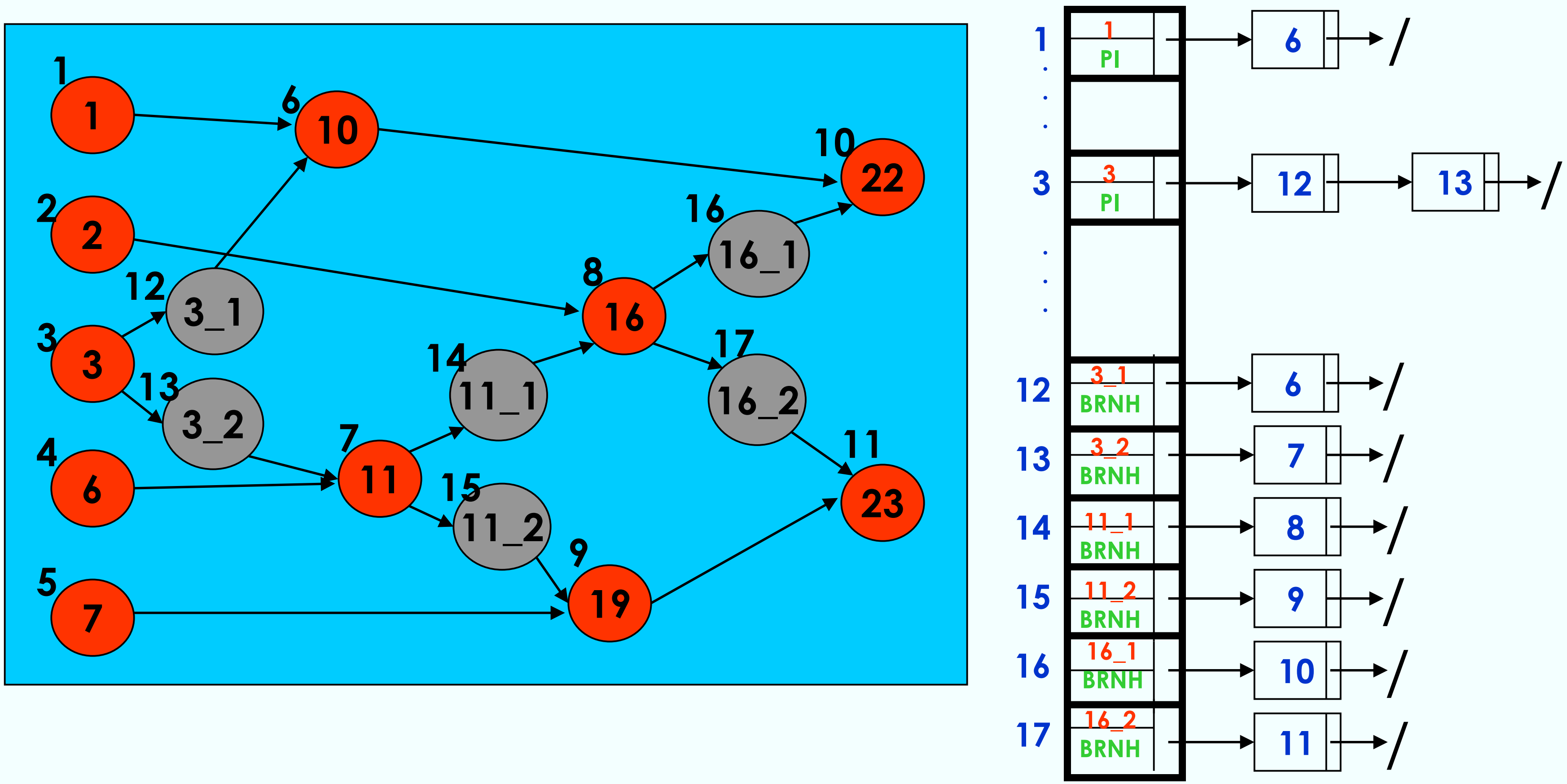
- The bench format does not *explicitly* define fanout stems and branches
- It is implied that:
  x = AND(a, b, c)
  y = NOT(a)
  z = OR(a, b)



- To add branch nodes to the graph, define a new node type BRNH, and:



Adding BRANCH nodes to the graph (example for circuit c17)



Circuit/Graph Path Counting

- Problem of counting the # of paths in a combinational circuit reduces to a *modified* topological traversal.

- Let node v with $S(v)$ = set of immediate predecessors of v. The # of paths up to v, defined by $p(v)$, is:

$$p(v) = \sum p(i),\ i \in S(v)$$

Graph Path Counting Algorithm

Input: $G(V,E)$ with PI = set of nodes with no predecessors, PO = set of nodes with no successors, and $S(v)$ = set of immediate successors of node v.

Output: # of paths in G

Algorithm Count_Paths(G)

Step 1: FOR (every $v \in V$)
IF ($v \in PI$) $p[v] = 1$; mark v as visited;
ELSE $p[v] = 0$; mark v as not-visited;

Step 2: Run topological traversal algorithm; modified such that when a node v is processed, calculate $p(v) = \Sigma p(i), i \in S(v)$

Step 3: Return $\Sigma p(v), v \in PO$