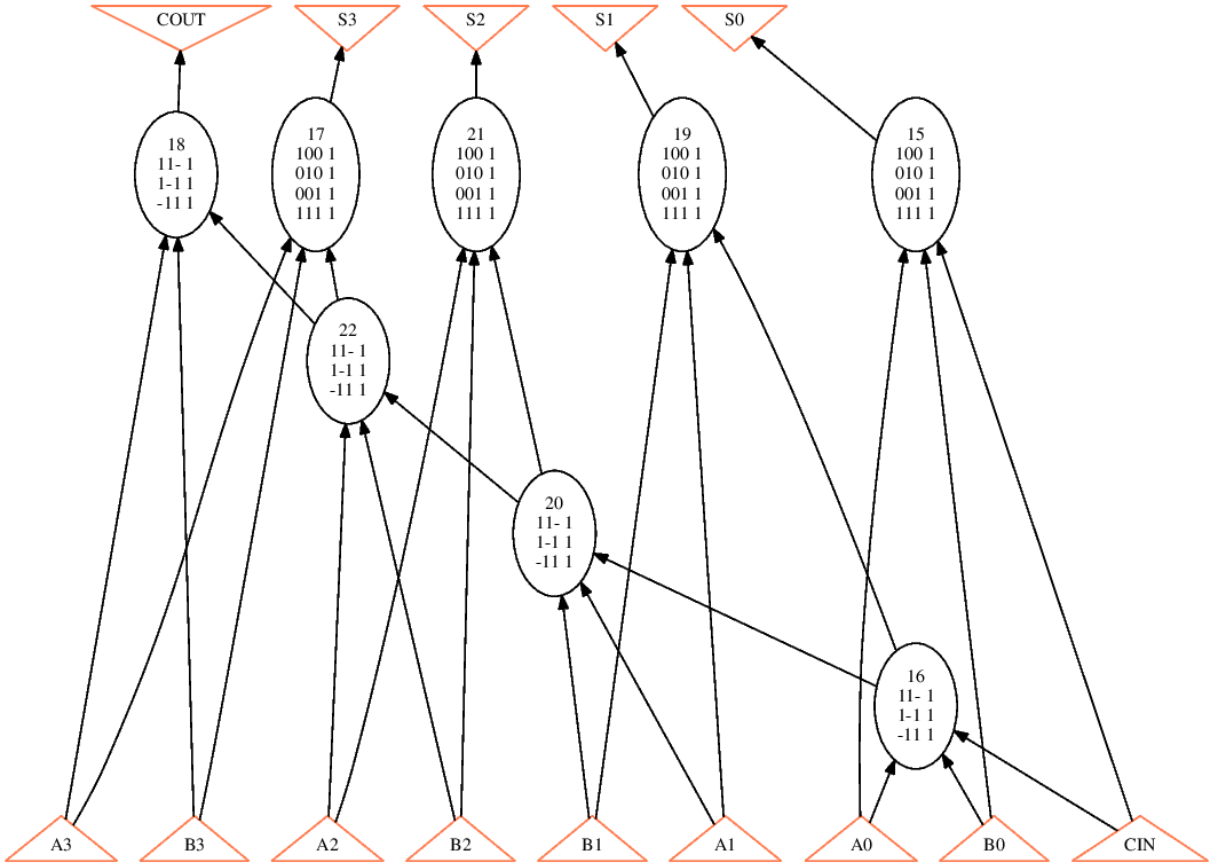
D08943012 謝承運 PA1

1. [Using ABC]

```
UC Berkeley, ABC 1.01 (compiled Sep 29 2020 10:20:02)
abc 01> read lsv/pa1/4_bit_adder.blif
Hierarchy reader flattened 4 instances of logic boxes and left 0 black boxes.
abc 02> print_stats
4_bit_adder               : i/o =    9/    5  lat =    0  nd =     8  edge =     24  cube =    28  lev = 4
abc 02> show
```

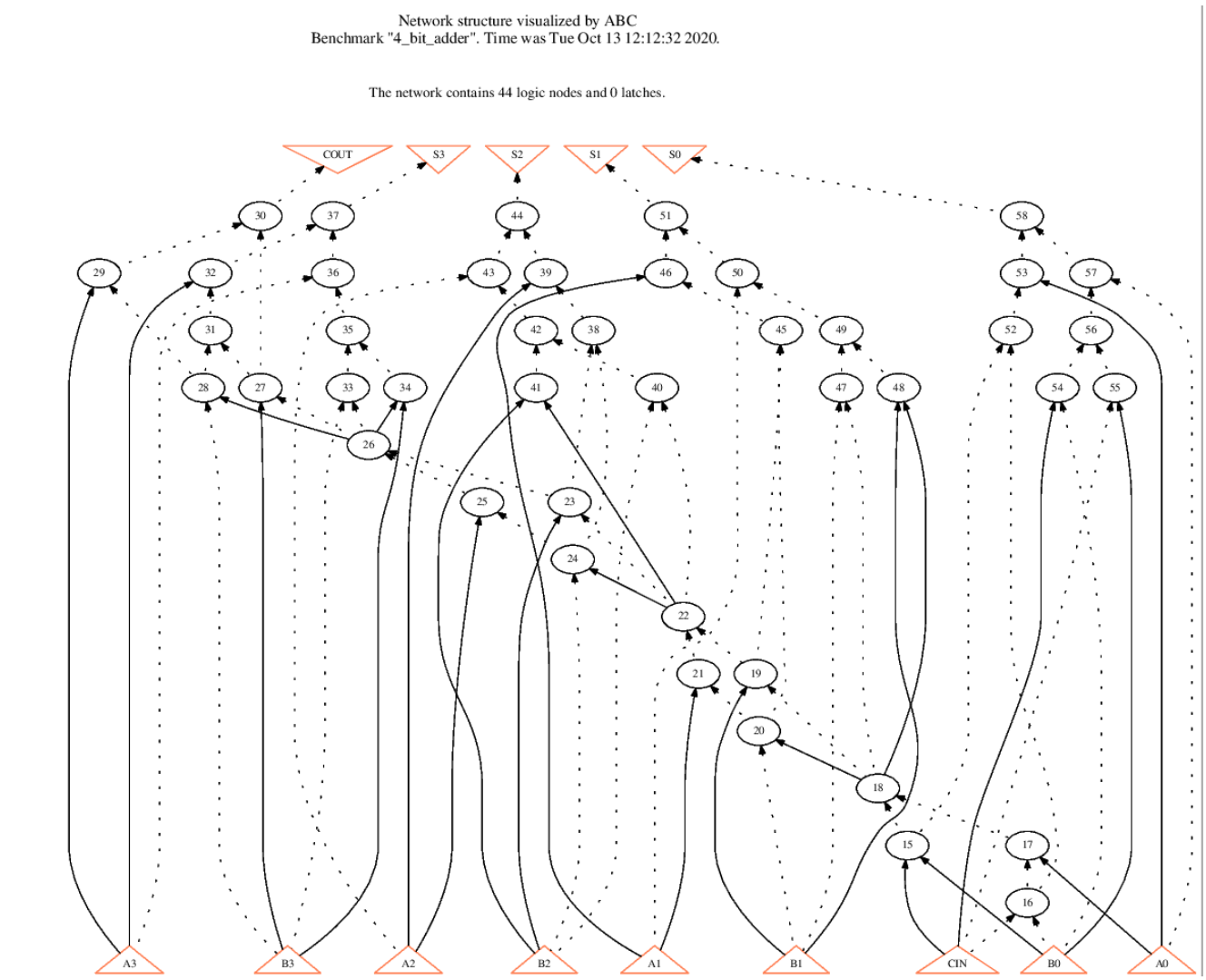
Network structure visualized by ABC
Benchmark "4_bit_adder". Time was Tue Oct 13 12:10:50 2020.

The network contains 8 logic nodes and 0 latches.

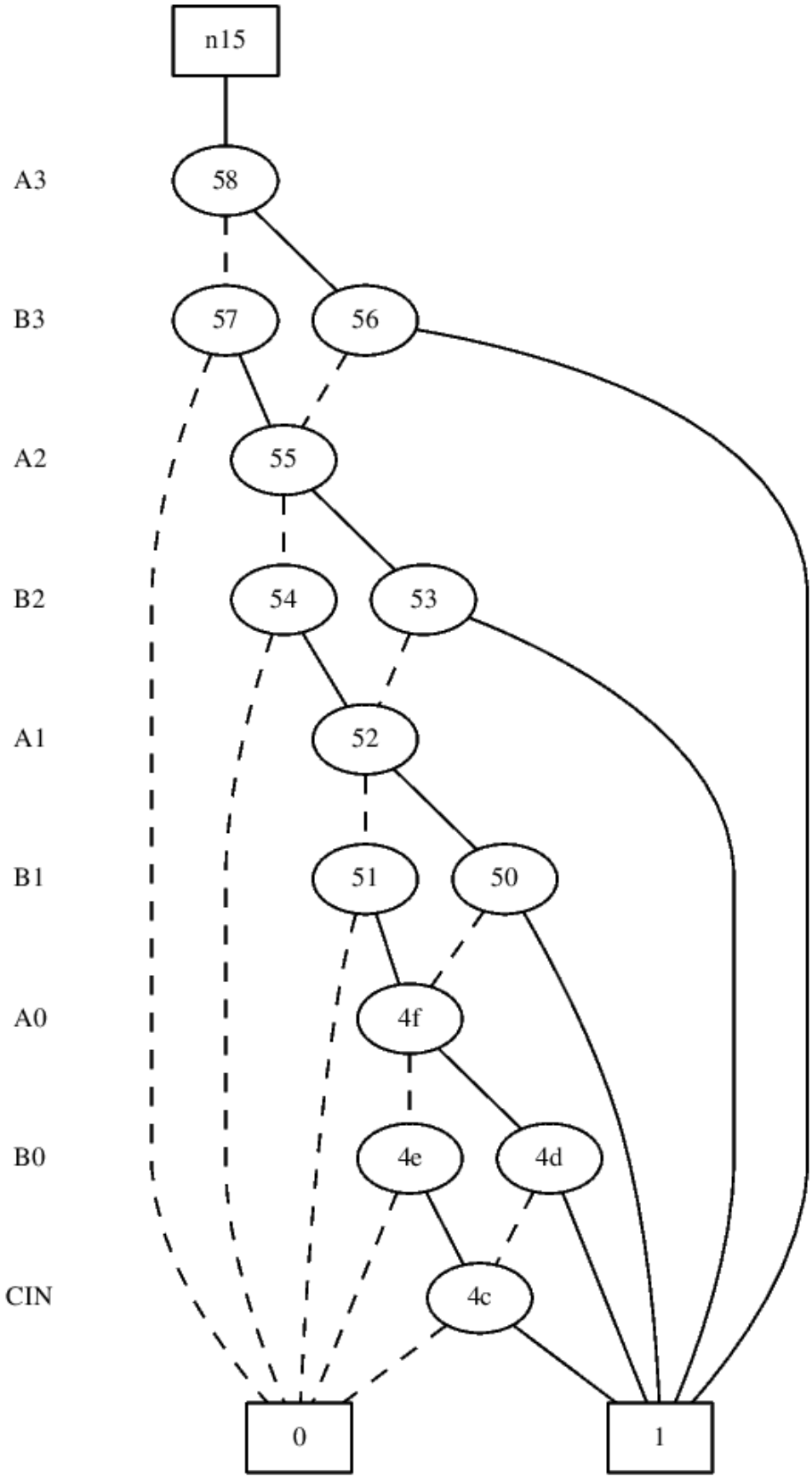


D08943012 謝承運 PA1

```
abc 03> strash
abc 04> show
```



```
abc 04> collapse
abc 05> show_bdd
```



2. [ABC Boolean Function Representations]

In ABC there are different ways to represent Boolean functions.

(a) Compare the following differences with the four-bit adder example.

1. logic network in AIG (by command aig) vs. structurally hashed AIG (by command strash)

```
abc 10> aig -h
usage: aig [-h]
                converts node functions to AIG
        -h     : print the command usage
abc 10> aig
abc 10> print_stats
4_bit_adder                : i/o =    9/    5  lat =    0  nd =     8  edge =     24  aig  =    52  lev = 4
```

command aig 只是將原始的 node 轉為 AIG 表示

```
abc 10> strash -h
usage: strash [-acrih]
                transforms combinational logic into an AIG
        -a    : toggles between using all nodes and DFS nodes [default = DFS]
        -c    : toggles cleanup to remove the dagling AIG nodes [default = all]
        -r    : toggles using the record of AIG subgraphs [default = no]
        -i    : toggles complementing the POs of the AIG [default = no]
        -h    : print the command usage
abc 10> strash
abc 11> print_stats
4_bit_adder                  : i/o =    9/    5  lat =    0  and =     44  lev = 13
```

command strash 則是將整個邏輯轉為 AIG

所以做完 command aig edge 數不變但多了一些 aig node，做完 command strash 則是剩下 and 與 inverters。

2. logic network in BDD (by command bdd) vs. collapsed BDD (by command collapse)

```
abc 13> bdd -h
usage: bdd [-rsh]
                converts node functions to BDD
        -r     : toggles enabling dynamic variable reordering [default = yes]
        -s     : toggles constructing BDDs directly from SOPs [default = no]
        -h     : print the command usage
abc 13> bdd
abc 13> print_stats
4_bit_adder                : i/o =    9/    5  lat =    0  nd =     8  edge =     24  bdd  =    28  lev = 4
```

command bdd 只是將原始的 node 轉為 BDD 表示

所以做完 command bdd edge 數不變但多了一些 bdd node

```
abc 14> collapse -h
usage: collapse [-B <num>] [-L file] [-rodxvh]
                 collapses the network by constructing global BDDs
        -B <num>: limit on live BDD nodes during collapsing [default = 1000000000]
        -L file : the log file name [default = no logging]
        -r      : toggles dynamic variable reordering [default = yes]
        -o      : toggles reverse variable ordering [default = no]
        -d      : toggles dual-rail collapsing mode [default = no]
        -x      : toggles dumping file "order.txt" with variable order [default = no]
        -v      : print verbose information [default = no]
        -h      : print the command usage
abc 14> collapse
abc 15> print_stats
4_bit_adder                : i/o =    9/    5  lat =    0  nd =     5  edge =     33  bdd  =    43  lev = 1
```

command collapse 則是將整個邏輯轉為一個 BDD

(b) Given a structurally hashed AIG, find a sequence of ABC command(s) to covert it to a logic network with node function expressed in sum-of-products (SOP).

1. strash
2. logic