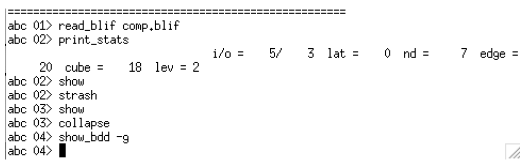
LSV PA1

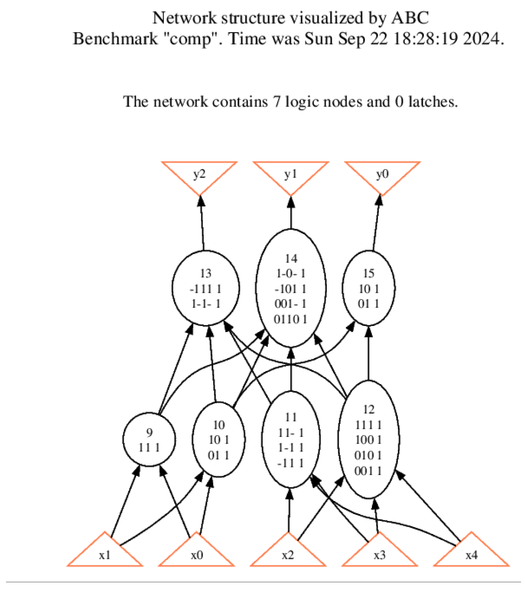
何睿濬 B11202008

2.using ABC

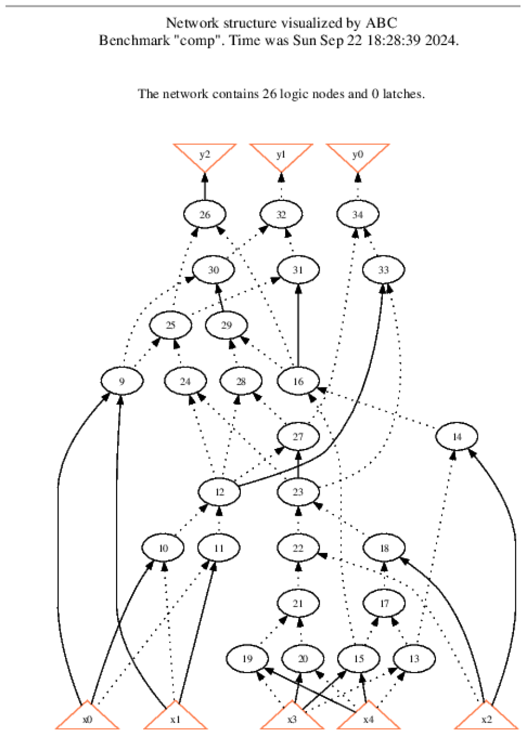


```
==================================================
abc 01> read_blif comp.blif
abc 02> print_stats
                                i/o =    5/    3  lat =    0  nd =     7  edge =
     20  cube =    18  lev = 2
abc 02> show
abc 02> strash
abc 03> show
abc 03> collapse
abc 04> show_bdd -g
abc 04> █
```

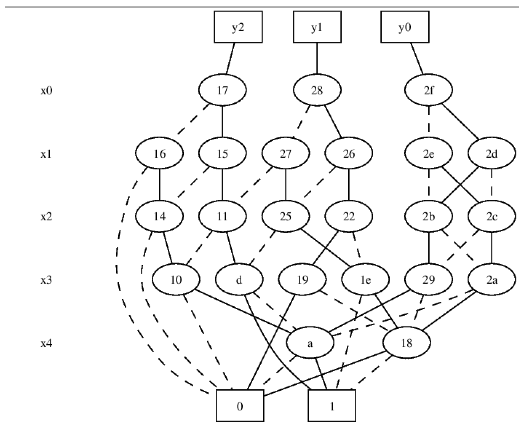
screenshot of terminal



screenshot of show original network


Network structure visualized by ABC
Benchmark "comp". Time was Sun Sep 22 18:28:39 2024.
The network contains 26 logic nodes and 0 latches.
y2
y1
y0
x0
x1
x3
x4
x2

screenshot of show strash

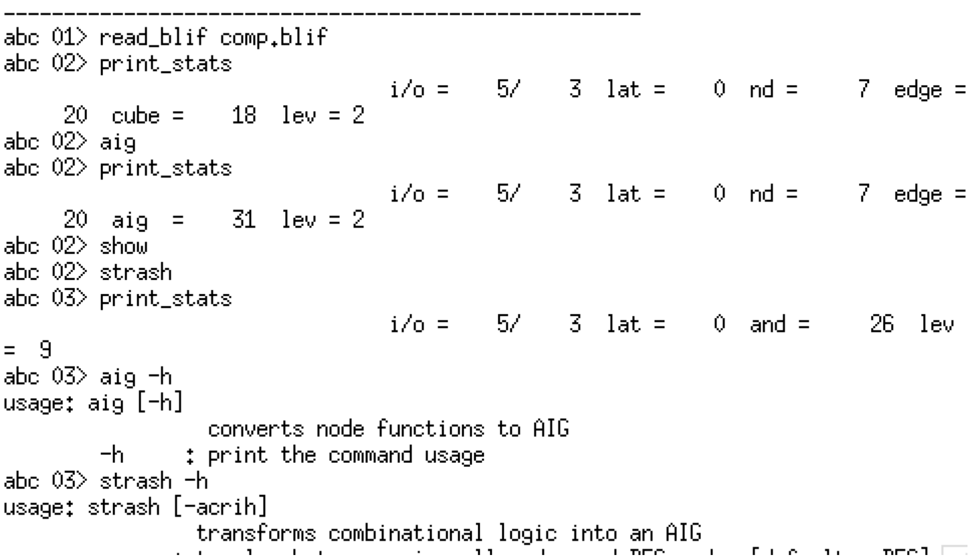


screenshot of show collapse

3.ABC Boolean Function Representations

(a)

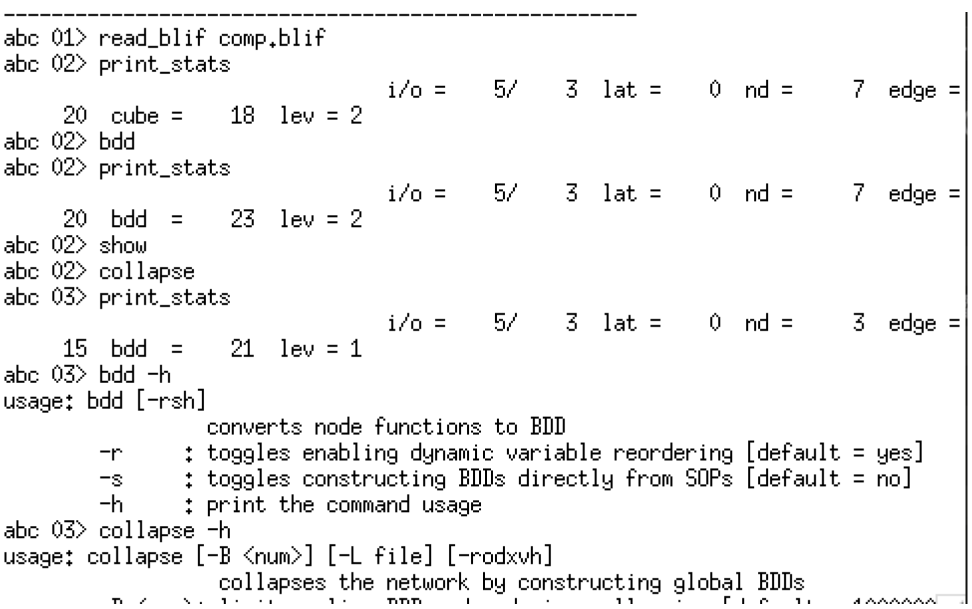
1.



```
--------------------------------------------------------
abc 01> read_blif comp.blif
abc 02> print_stats
                                i/o =    5/    3  lat =    0  nd =     7  edge =
     20  cube =    18  lev = 2
abc 02> aig
abc 02> print_stats
                                i/o =    5/    3  lat =    0  nd =     7  edge =
     20  aig  =    31  lev = 2
abc 02> show
abc 02> strash
abc 03> print_stats
                                i/o =    5/    3  lat =    0  and =     26  lev
=  9
abc 03> aig -h
usage: aig [-h]
                 converts node functions to AIG
        -h     : print the command usage
abc 03> strash -h
usage: strash [-acrih]
                transforms combinational logic into an AIG
```

From the graph above, we can found that the graph have the same number of node after running command 'aig', but the functions in every node are transformed into AIG form, and command 'strash' break all nodes and change the whole graph into AIG.
We can also get the same conclusion through help message. 'aig' only converts node function to AIG, and 'strash' transform the whole logic network into AIG.

2.

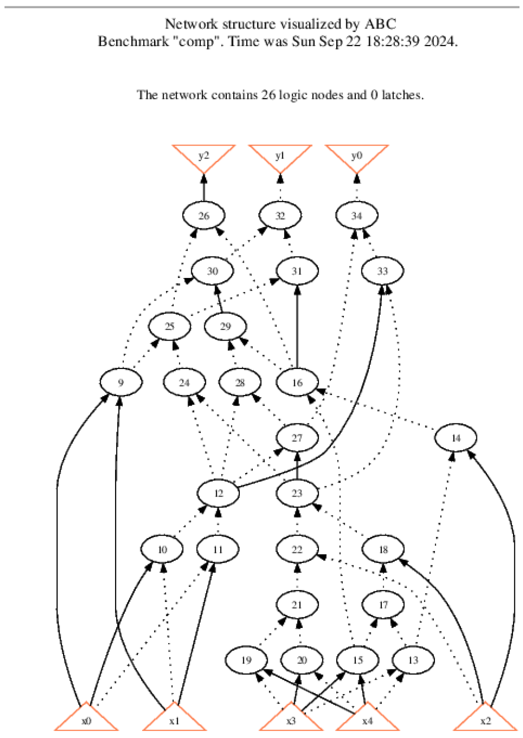


```
--------------------------------------------------------
abc 01> read_blif comp.blif
abc 02> print_stats
                                i/o =    5/    3  lat =    0  nd =     7  edge =
     20  cube =    18  lev = 2
abc 02> bdd
abc 02> print_stats
                                i/o =    5/    3  lat =    0  nd =     7  edge =
     20  bdd  =    23  lev = 2
abc 02> show
abc 02> collapse
abc 03> print_stats
                                i/o =    5/    3  lat =    0  nd =     3  edge =
     15  bdd  =    21  lev = 1
abc 03> 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 03> collapse -h
usage: collapse [-B <num>] [-L file] [-rodxvh]
                  collapses the network by constructing global BDDs
```

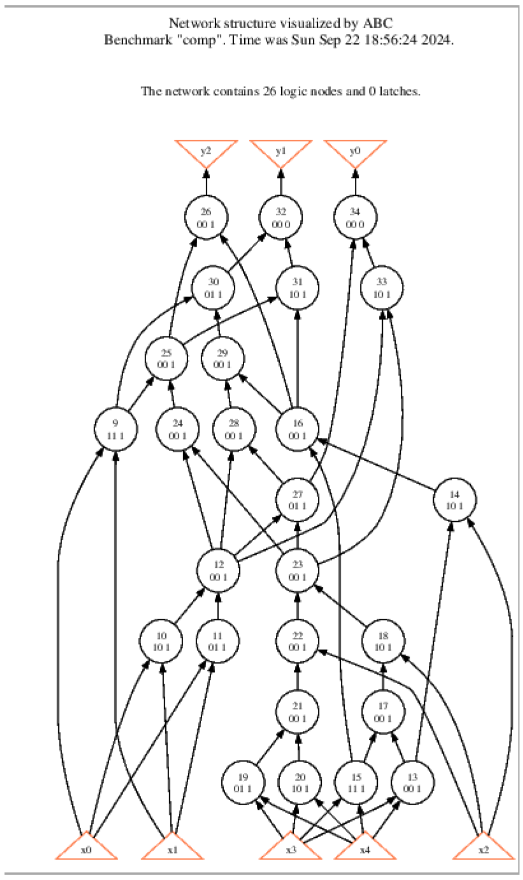
Similar to difference between 'aig' and 'strash', 'bdd' transform only node funciotn into BDD form, and 'collapse' transform the whole logic network into BDD, and then use it to collapse the whole network into one level.

(b)

From the help message of command 'logic', we can know that it transforms an AIG into a logic network with SOPs, hence we can done it by simply type command 'logic', and the result is as following graph.


Network structure visualized by ABC
Benchmark "comp". Time was Sun Sep 22 18:28:39 2024.
The network contains 26 logic nodes and 0 latches.
y2
y1
y0
x0
x1
x3
x4
x2

structurally hashed AIG


Network structure visualized by ABC
Benchmark "comp". Time was Sun Sep 22 18:56:24 2024.
The network contains 26 logic nodes and 0 latches.
y2
y1
y0
x0
x1
x3
x4
x2

after typing 'logic'