LSV PA1 Report

r13943063 陳立強

2 [Using ABC]

(b)

1. read the BLIF file into ABC (command “read”)

```
abc 01> read comp.blif
abc 02>
```

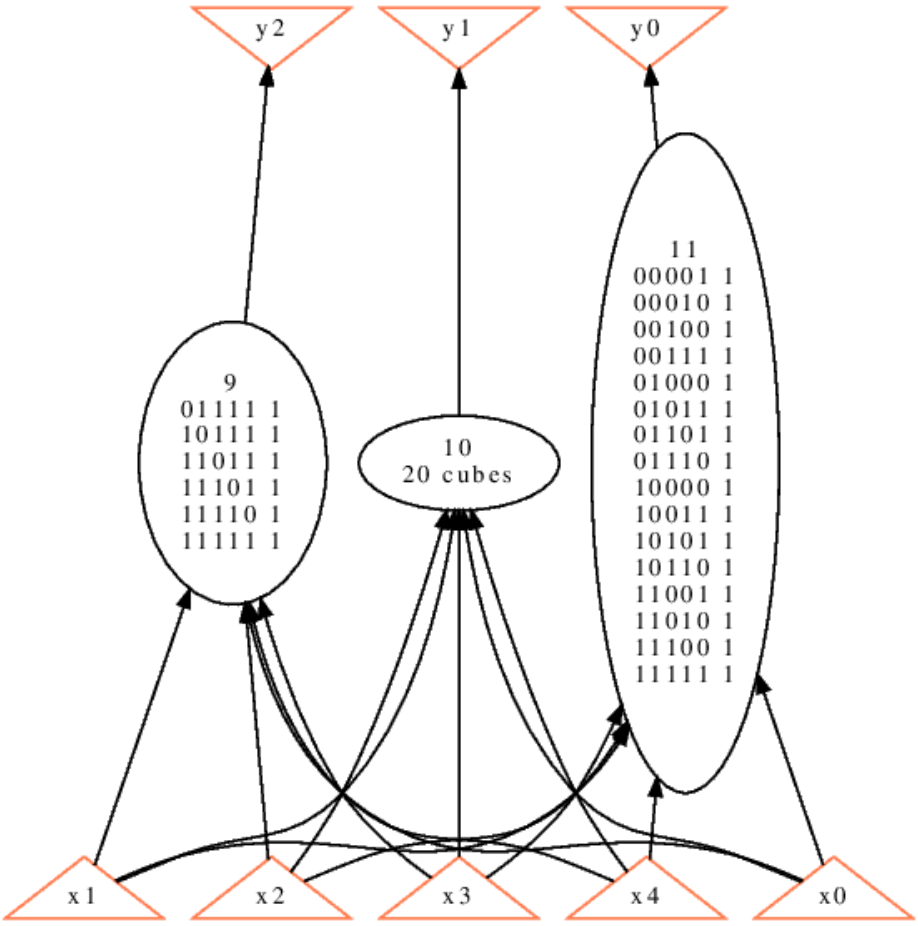
2. check statistics (command “print stats”)

```
abc 02> print_stats
comp                  : i/o =    5/    3  lat =    0  nd =     3  edge =     15  cube =    42  lev = 1
```

3. visualize the network structure (command “show”)

Network structure visualized by ABC
Benchmark "comp". Time was Fri Sep 13 21:44:12 2024.

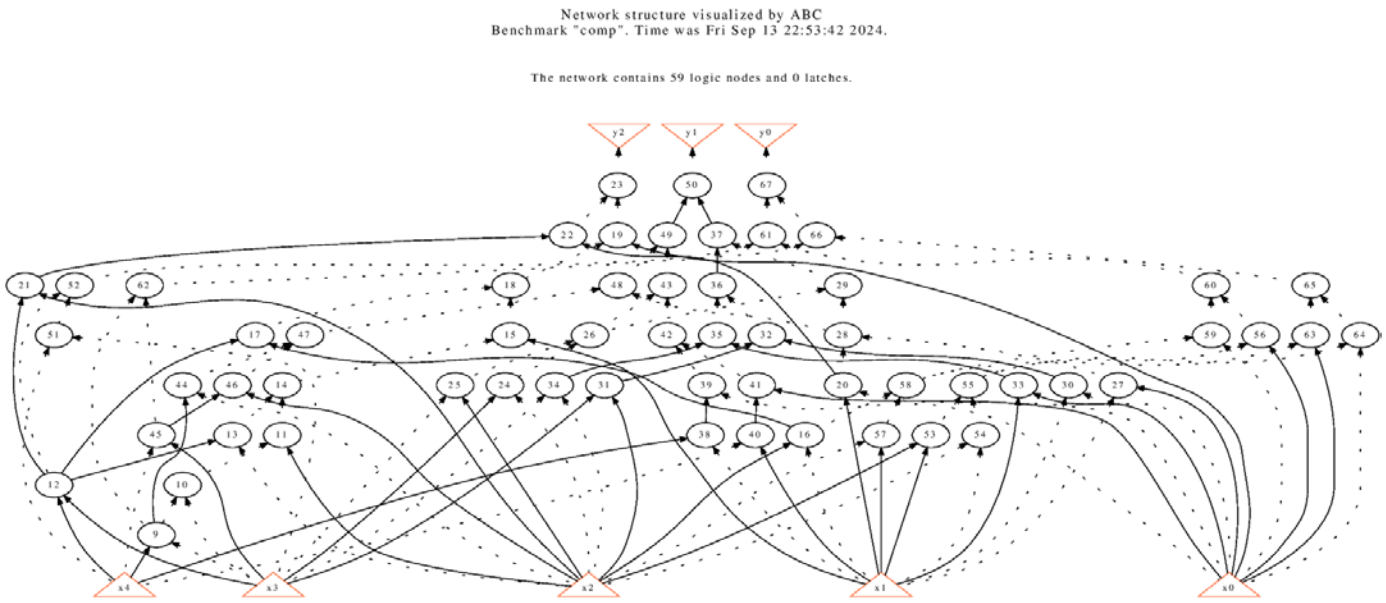
The network contains 3 logic nodes and 0 latches.



4. convert to AIG (command “strash”)

```
abc 02> strash
abc 03>
```

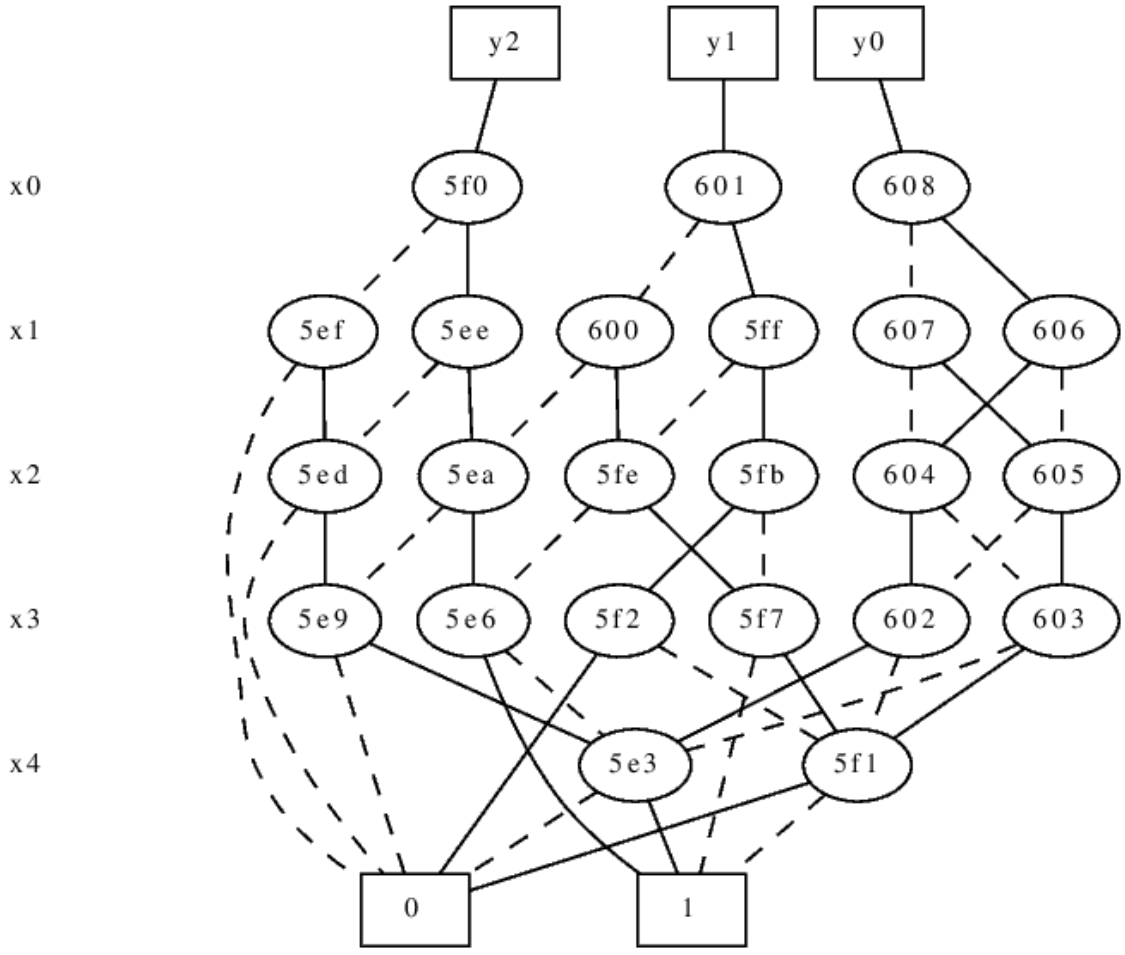
5. visualize the AIG (command “show”)



6. convert to BDD (command “collapse”)

```
abc 03> collapse
abc 04>
```

7. visualize the BDD (command “show bdd -g”; note that “show bdd” only shows the first PO; option “-g” can be applied to show all POs)

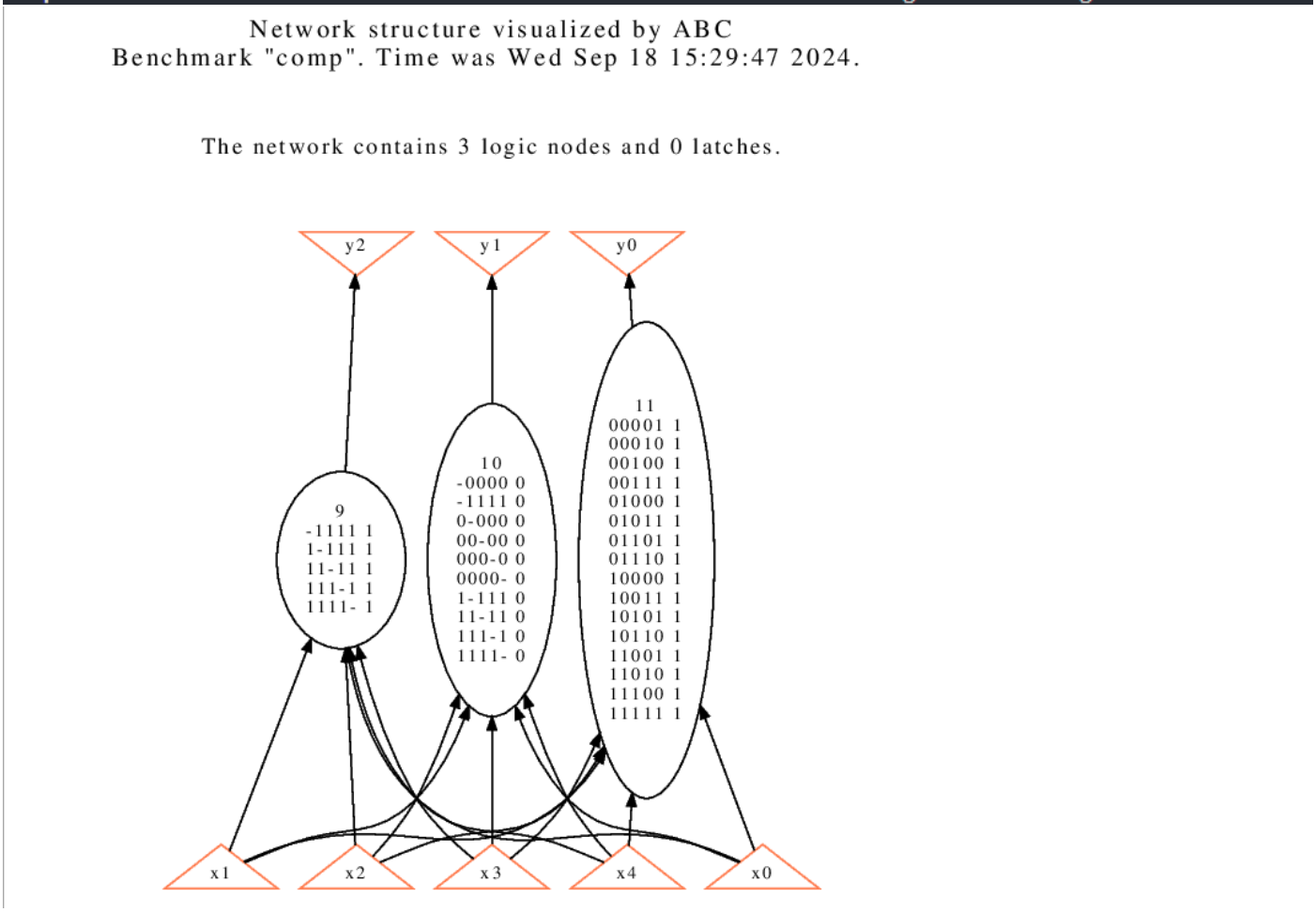


3 [ABC Boolean Function Representations]

(a) 1.

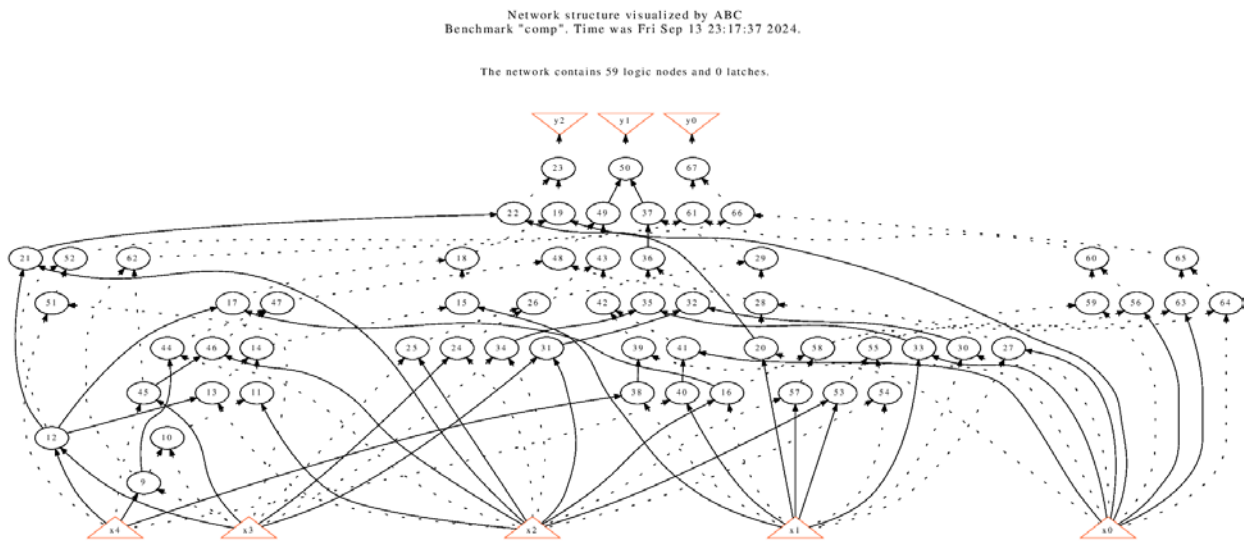
By command “aig”, we can see that the network structure showed below is still the same as the original one, this command converts node functions to AIG.

```
abc 02> aig
abc 02> print_stats
comp                  : i/o =    5/    3  lat =    0  nd =     3  edge =     15  aig  =    65  lev = 1
```



By command “strash”, it transforms combinational logic into an AIG, with structurally hashed.

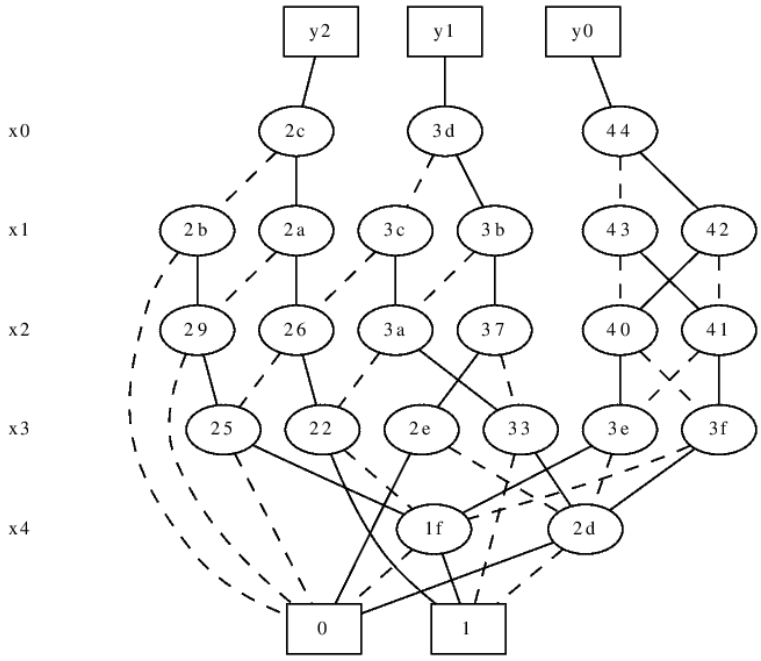
```
strash
abc 03> print_stats
comp                  : i/o =    5/    3  lat =    0  and =    59  lev =  8
```



2.

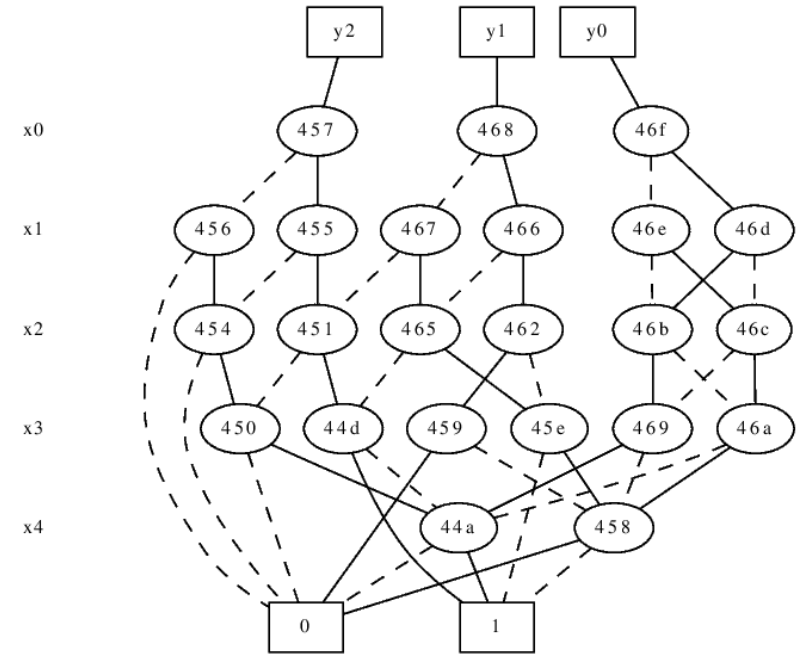
By command “bdd”, we can see that the result is the same as command “collapse”.

```
abc 02> bdd
abc 02> print_stats
comp                    : i/o =    5/    3  lat =    0  nd =     3  edge =     15  bdd  =    21  lev = 1
```



By command “collapse”, we can see that the result is the same as command “bdd”.

```
collapse
abc 03> print_stats
comp                    : i/o =    5/    3  lat =    0  nd =     3  edge =     15  bdd  =    21  lev = 1
```



(b)

Use command “logic” to transforms the AIG into a logic network with the SOP representation of the two-input AND-gates.

```
abc 01> read_blif comp.blif
abc 02> strash
abc 03> print_stats
comp                          : i/o =    5/    3  lat =    0  and =     59  lev =  8
abc 03> logic
abc 04> print_stats
comp                          : i/o =    5/    3  lat =    0  nd =    59  edge =    118  cube =    59  lev = 8
abc 04> show
```