Logic Synthesis & Verification, Fall 2024
Programming Assignment 1

R13943081 藍苡芯

2. (b)

```
abc 01> read ../HW/lsv/pa1/comp.blif
abc 02> print_stats
5to3_compressor             : i/o =    5/    3  lat =    0  nd =     3  edge =     15  cube =    42  lev = 1
abc 02> show
abc 02> strash
abc 03> show
abc 03> collapse
abc 04> show_bdd -g
```

Figure 1: Screenshot of running commands

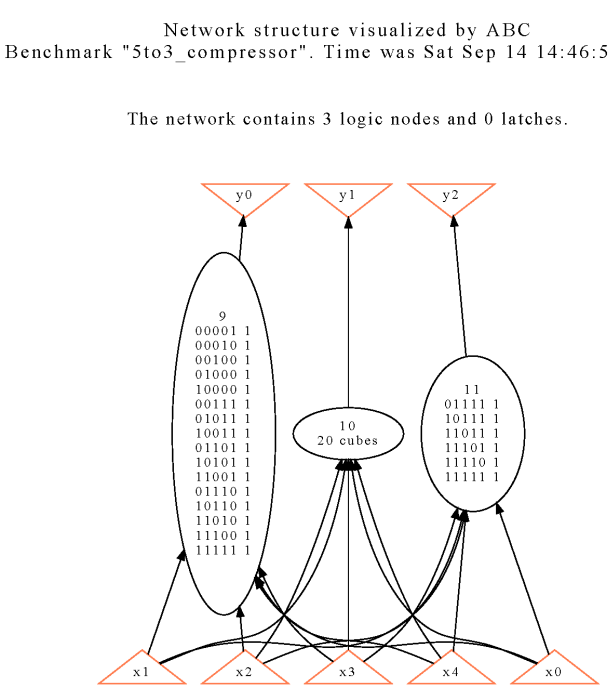

Figure 2: Visualization of the network structure

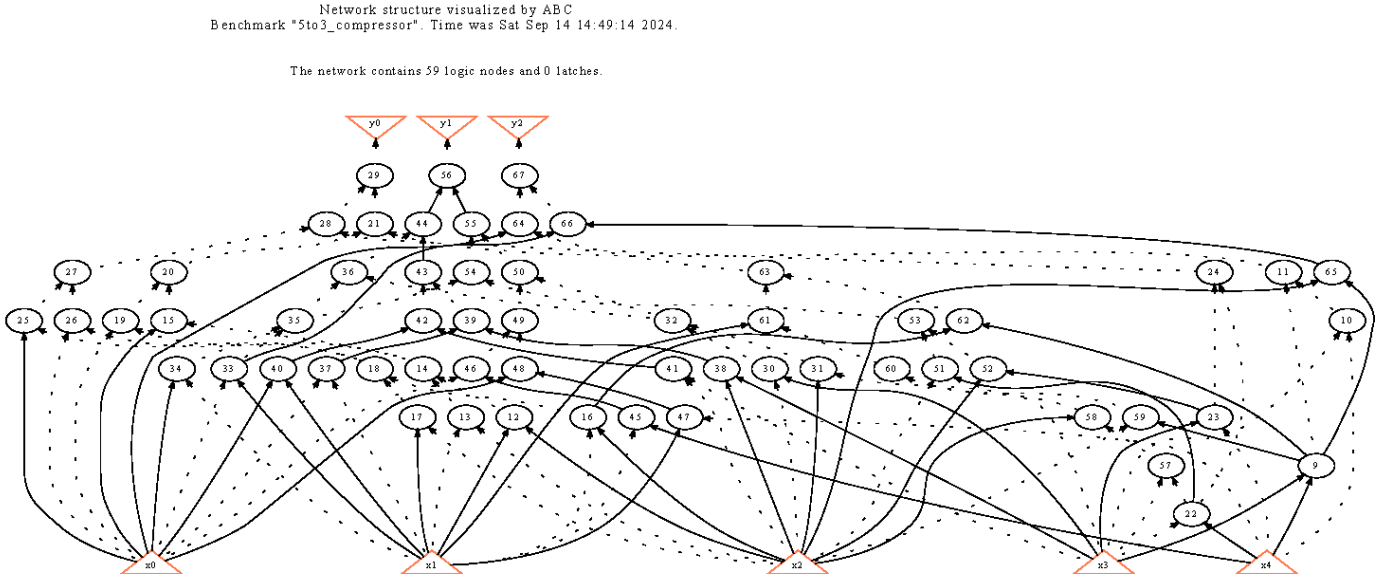

Figure 3: Visualization of the AIG

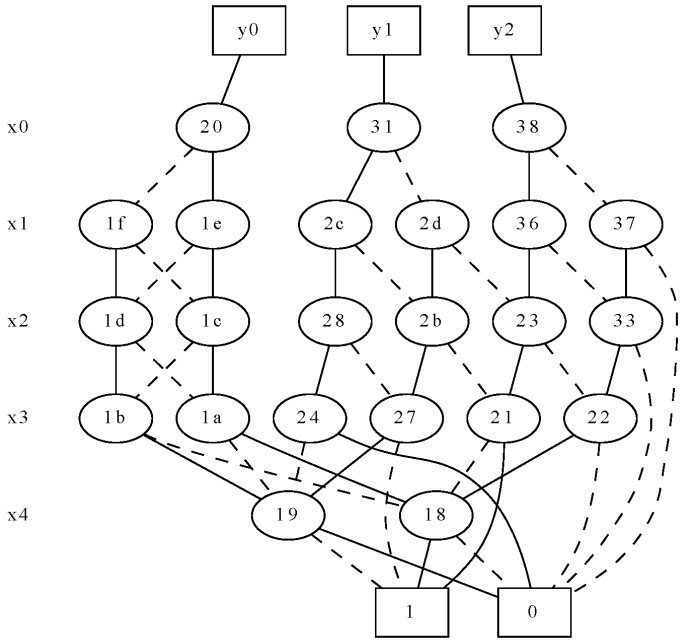

Figure 4: Visualization of the BDD

3. (a)
   1. command ”aig” vs. command ”strash”

```
abc 01> read ../HW/lsv/pa1/comp.blif
abc 02> aig
abc 02> print_stats
5to3_compressor             : i/o =    5/    3  lat =    0  nd =     3  edge =     15  aig  =    65  lev = 1
abc 02> show
abc 02>  read ../HW/lsv/pa1/comp.blif
abc 03> strash
abc 04> print_stats
5to3_compressor             : i/o =    5/    3  lat =    0  and =     59  lev =  8
```

Figure 5: Logic network in AIG vs. structurally hashed AIG

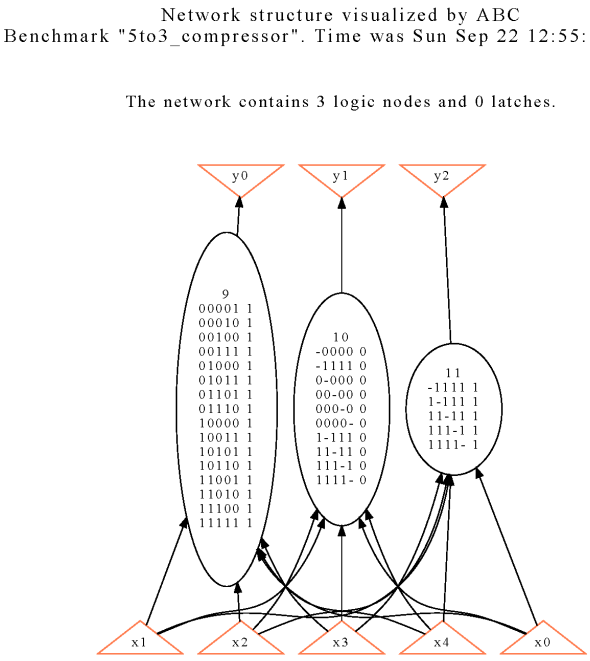

Figure 6: Visualization using ”aig” command

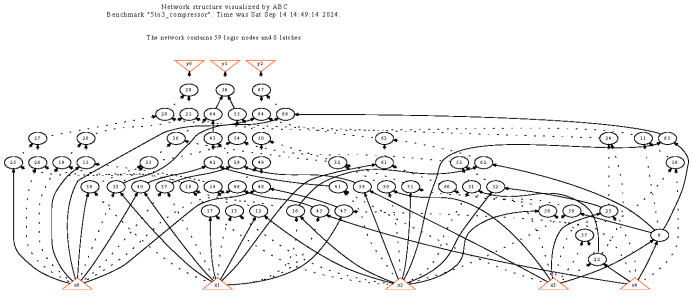

Figure 7: Visualization using ”strash” command

From fig. 5, it is observed that using the ”aig” command results in fewer nodes compared to the ”strash” command. This occurs because ”strash” transforms the network into an AIG by one-level structural hashing, simplifying the overall structure of the logic network.
On the other hand, ”aig” command converts local functions of the nodes to AIGs, resulting in the visualization using ”aig” command would have similar structures with directly reading BLIF file, while ”strash” produces a different visualization.

2. command ”bdd” vs. command ”collapse”

```
abc 01>  read ../HW/lsv/pa1/comp.blif
abc 02> bdd
abc 02> print_stats
5to3_compressor            : i/o =    5/    3  lat =    0  nd =     3  edge =     15  bdd  =    21  lev = 1
abc 02>  read ../HW/lsv/pa1/comp.blif
abc 03> collapse
abc 04> print_stats
5to3_compressor            : i/o =    5/    3  lat =    0  nd =     3  edge =     15  bdd  =    21  lev = 1
```

Figure 8: logic network in BDD vs. collapsed BDD

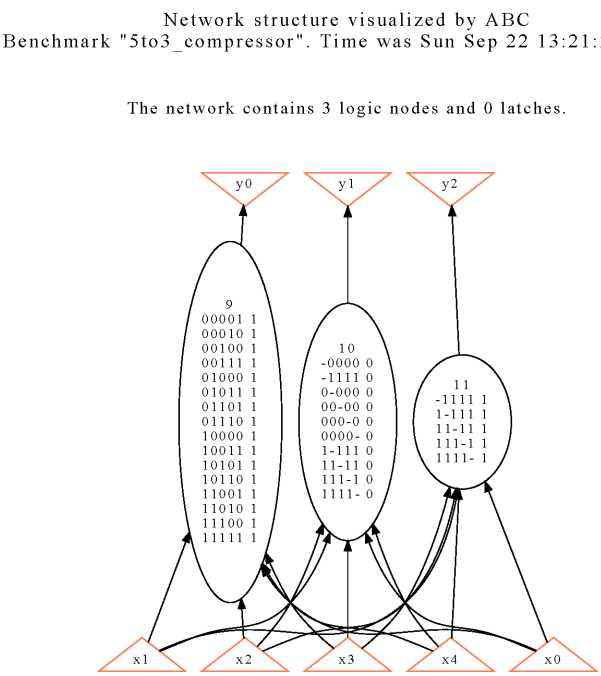

Figure 9: logic network in BDD vs. collapsed BDD

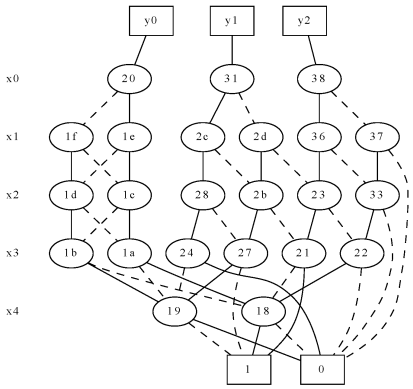

Figure 10: logic network in BDD vs. collapsed BDD

From fig. 8, it is observed that the ”bdd” command produces the same number of nodes as the ”collapse” command, making the two appear similar. However, ”collapse” builds global functions using BDDs, so it may simply the overall structure of the logic network.
On the other hand, ”bdd” command converts local functions of the nodes to BDDs, resulting in the visualization using ”bdd” command would have similar structures with directly reading BLIF file, while ”collapse” produces a different visualization.

(b)
The SOP form can be converted by the commands:

read; strash; logic

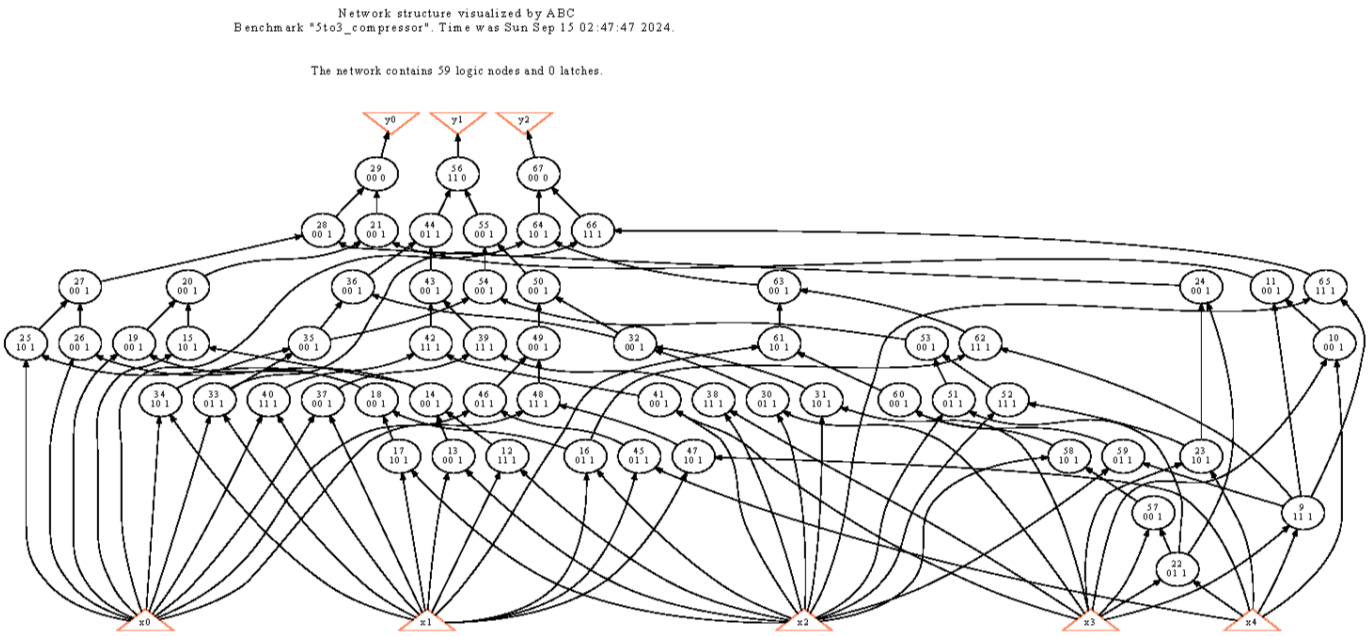

Figure 11: Visualization of the network in SOP

```
abc 01> read ../HW/lsv/pa1/comp.blif
abc 02> print_stats
5to3_compressor              : i/o =    5/    3  lat =    0  nd =     3  edge =     15  cube =    42  lev = 1
abc 02> strash
abc 03> print_stats
5to3_compressor              : i/o =    5/    3  lat =    0  and =     59  lev =  8
abc 03> logic
abc 04> print_stats
5to3_compressor              : i/o =    5/    3  lat =    0  nd =    59  edge =    118  cube =    59  lev = 8
```

Figure 12: Screenshot of running SOP conversion commands

The network structure is shown in fig. 11. As we can see, each internal node has same expression as SOP. Additionally, the states in different representations are shown in fig. 10.