Problem 2(b)

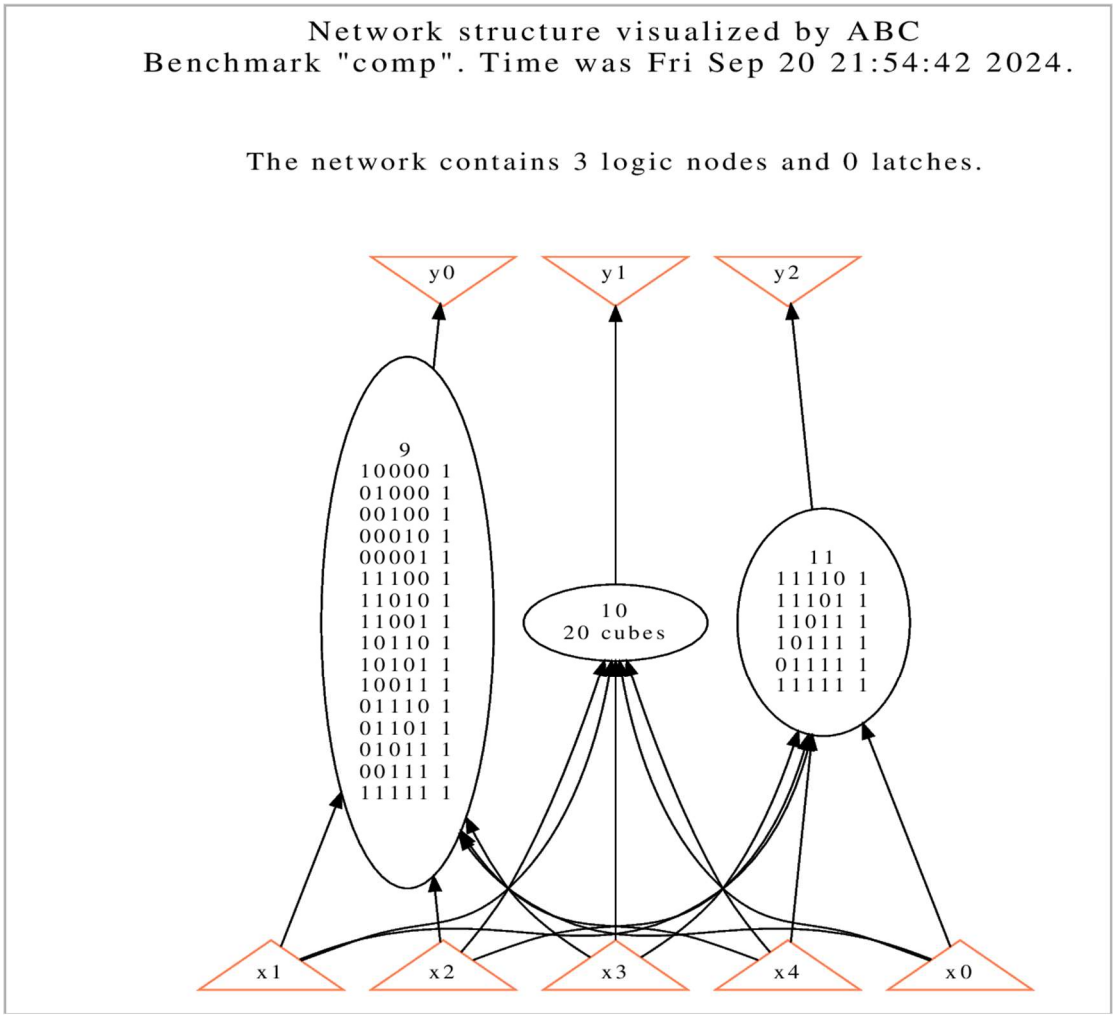
1.

```
abc 01> read ./lsv/pa1/comp.blif
abc 02>
```

2.

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

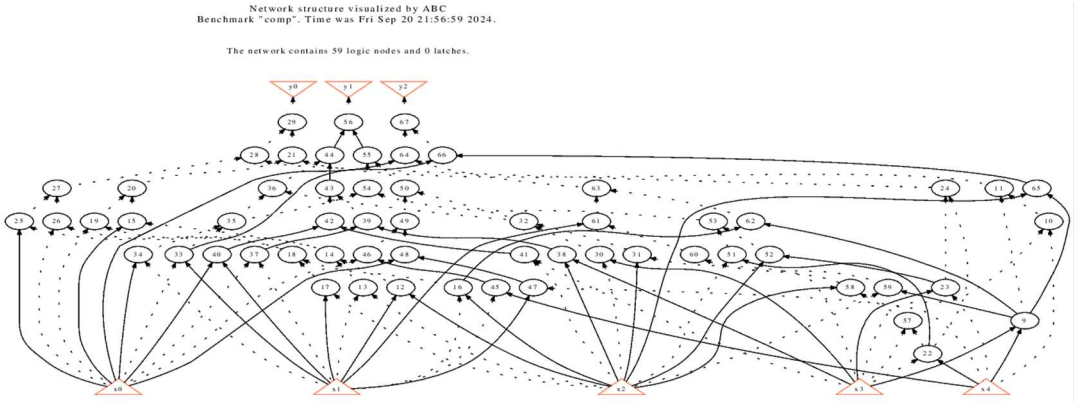
3.



4.

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

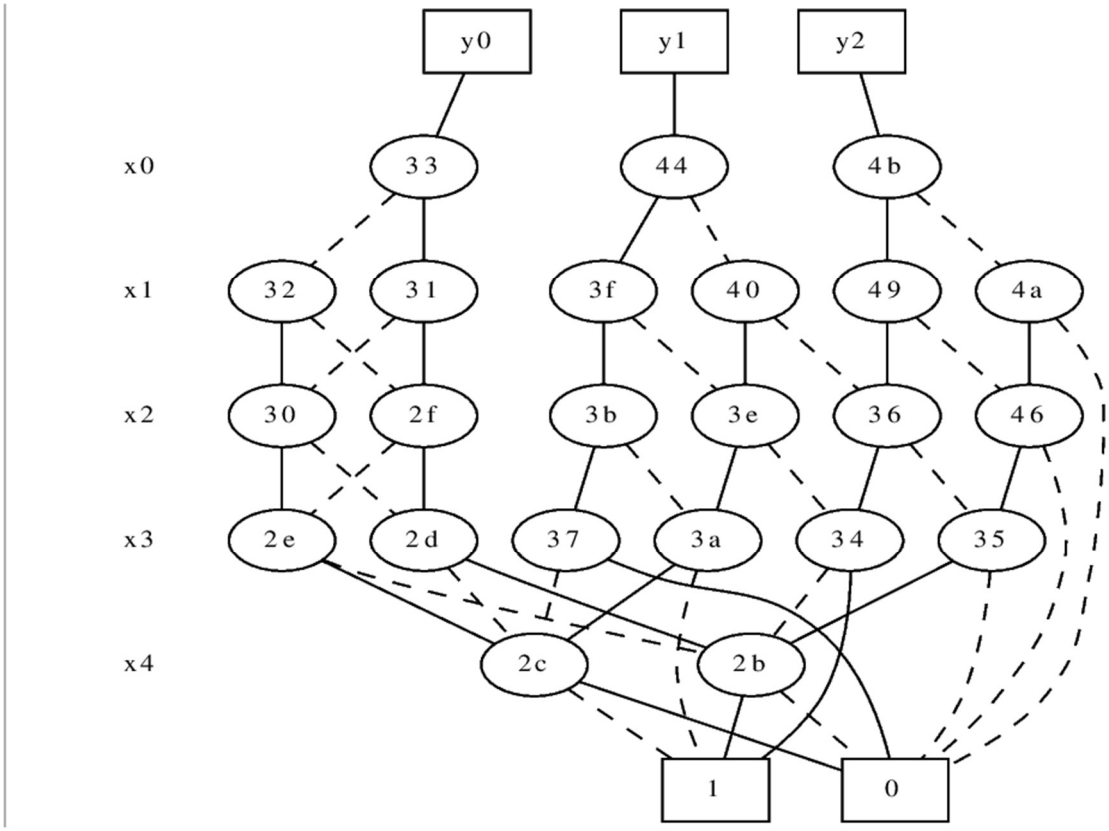
5.



6.

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

7.

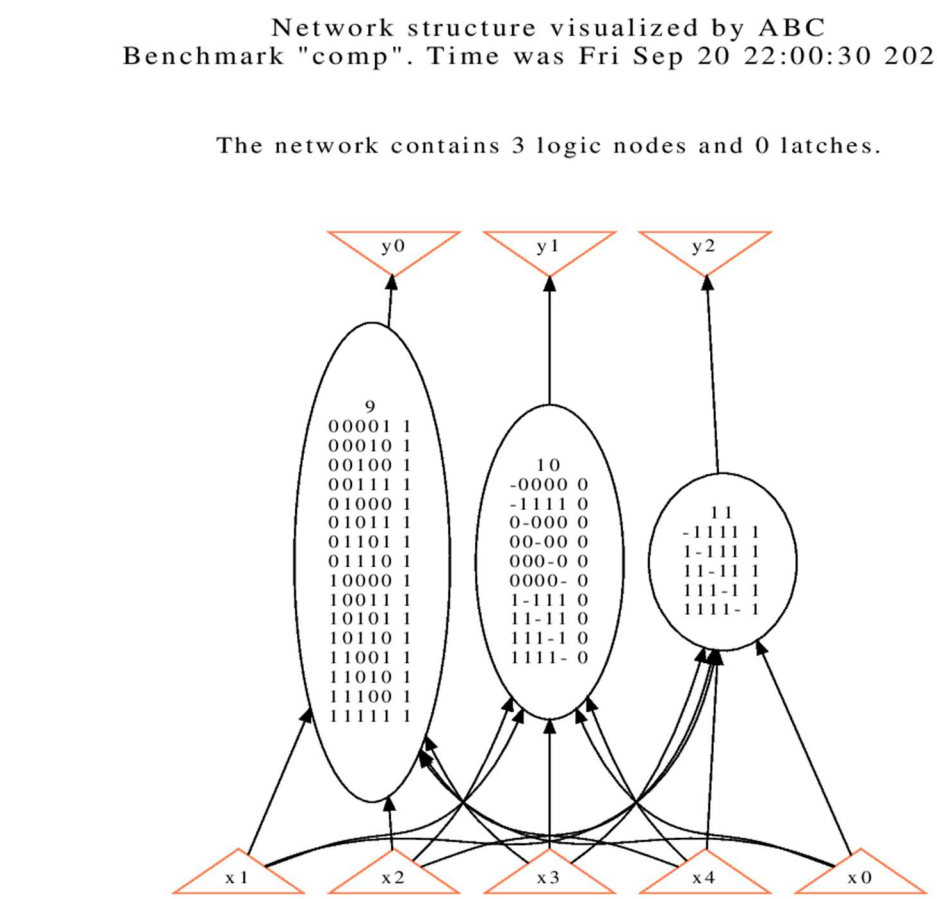


Problem 3(a)

1.

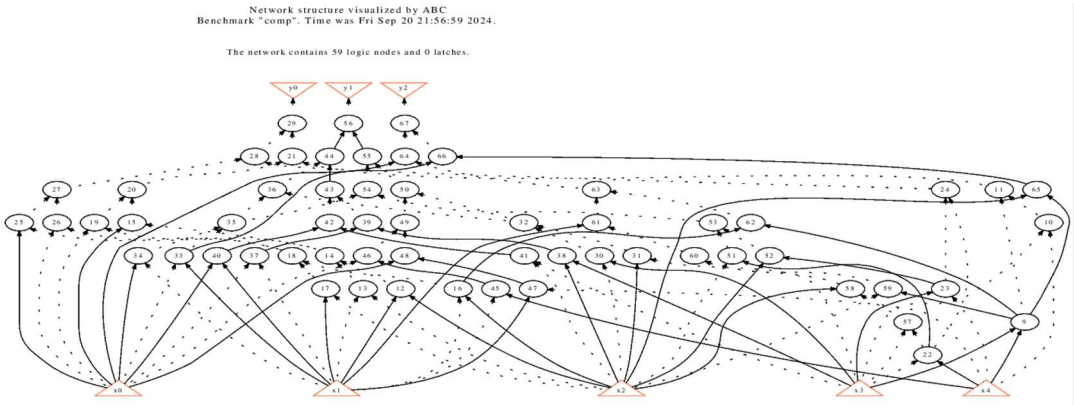
In command ‘aig’,

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



In command strash,

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

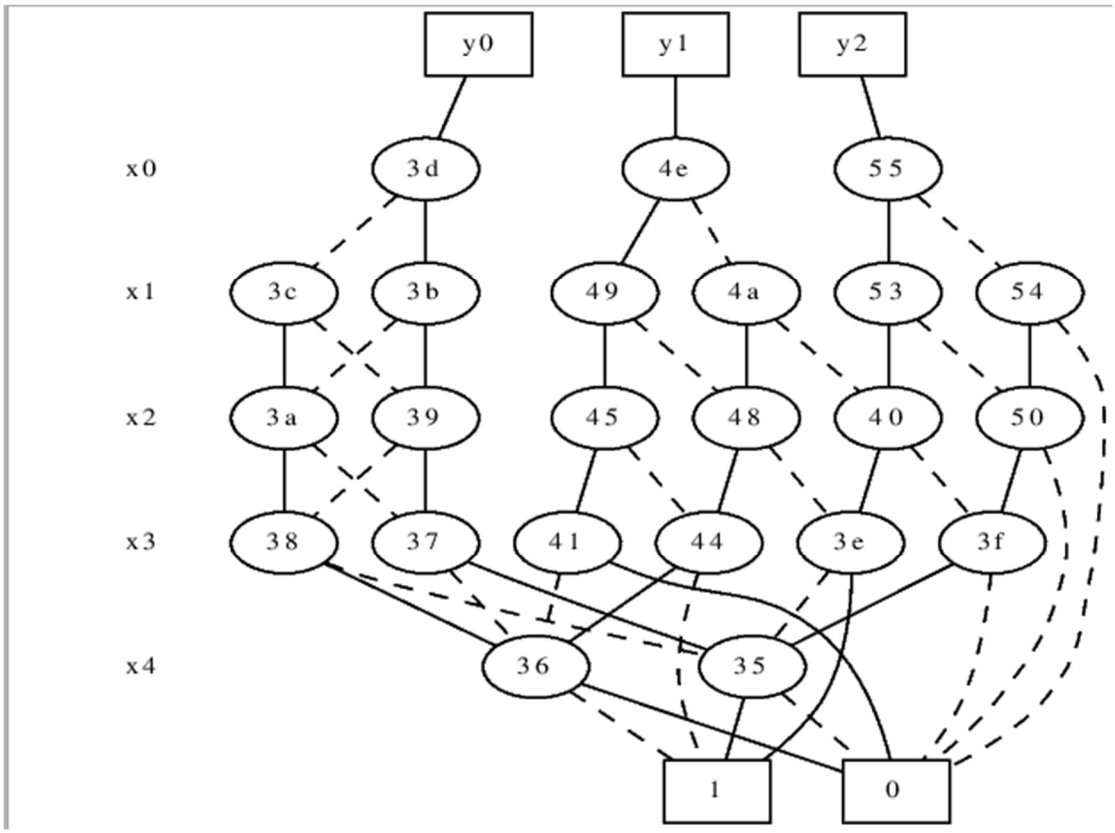


As shown in the above figures, the command ‘aig’ contains 65 aig components and only 1 level. At the contrast, the command ‘strash’ contains 59 and components and up to 8 levels. To sum up, the latter produces a more concise structure than the former at the expense of the level of the network is highly increased.

2.

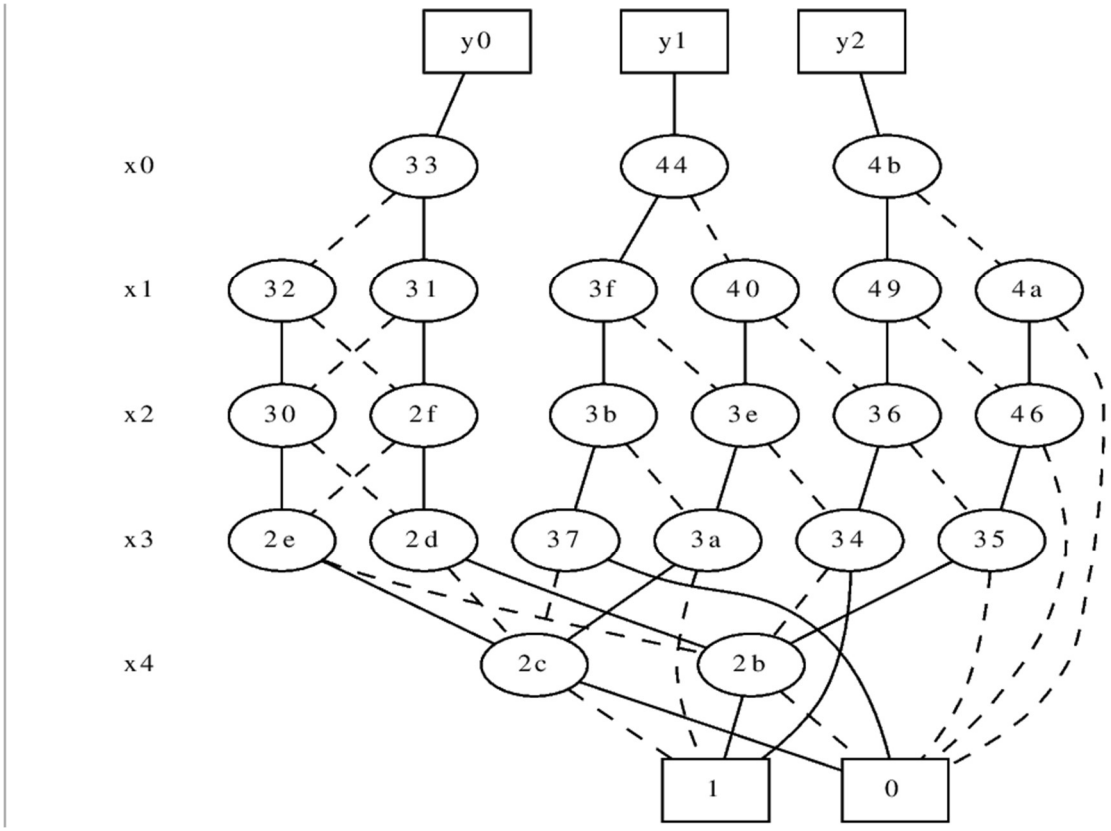
In command ‘bdd’,

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



In command ‘collapse’,

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



The two command seem to produce the same outcomes.

Problem 3(b)

```
abc 01> read ./lsv/pa1/comp.blif
abc 02> strash
abc 03> logic
abc 04> show
```

There is no difference if we continue to enter the command ‘sop’.