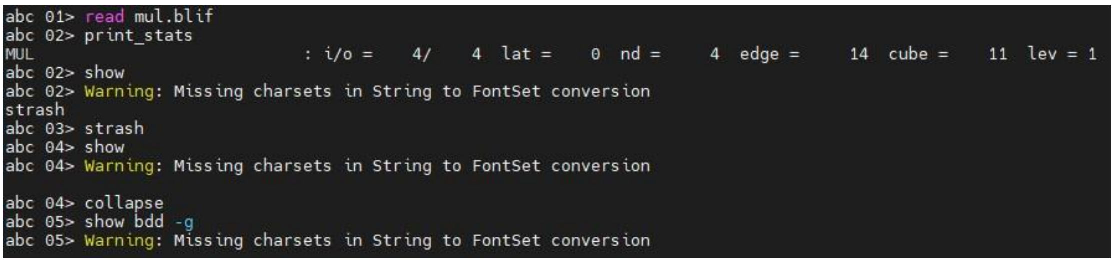
LSV PA1

R11943170 柯昱豪

2. [using ABC]

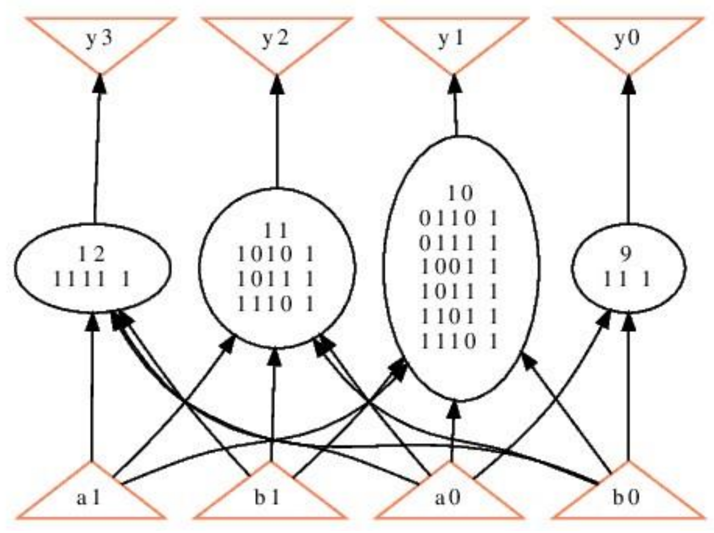
[commands]



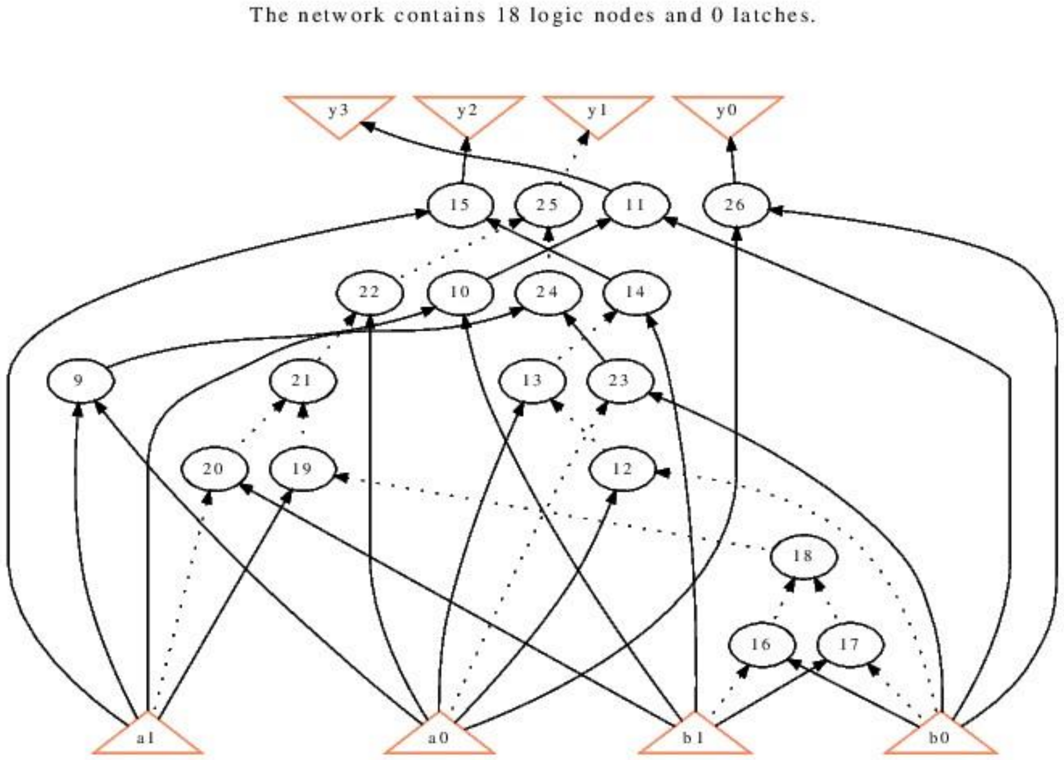
```
abc 01> read mul.blif
abc 02> print_stats
MUL                          : i/o =    4/    4  lat =    0  nd =     4  edge =     14  cube =     11  lev = 1
abc 02> show
abc 02> Warning: Missing charsets in String to FontSet conversion
strash
abc 03> strash
abc 04> show
abc 04> Warning: Missing charsets in String to FontSet conversion

abc 04> collapse
abc 05> show bdd -g
abc 05> Warning: Missing charsets in String to FontSet conversion
```

[visualize the network structure] (the first “show”)

The network contains 4 logic nodes and 0 latches.

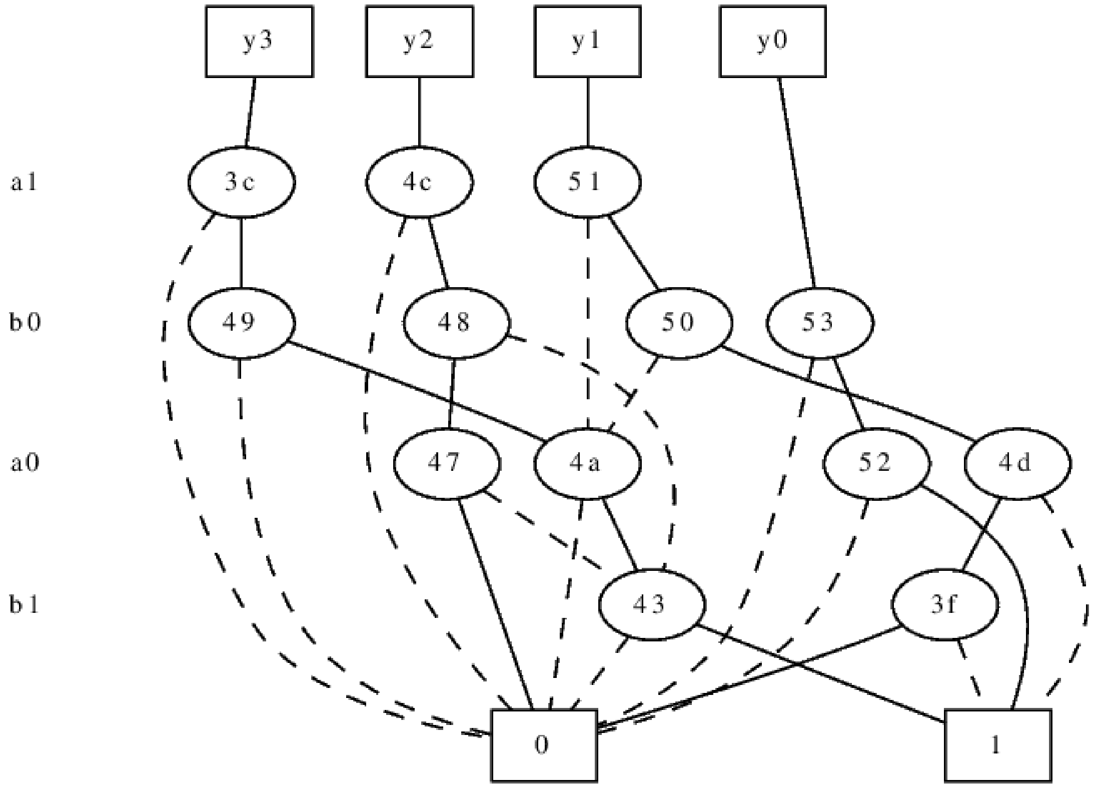


[visualize the AIG] (“show” after “strash”)



The network contains 18 logic nodes and 0 latches.

[visualize the BDD] (“show_bdd -g” after “collapse”)



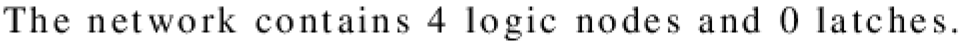
3. [ABC boolean function representations]

(a)

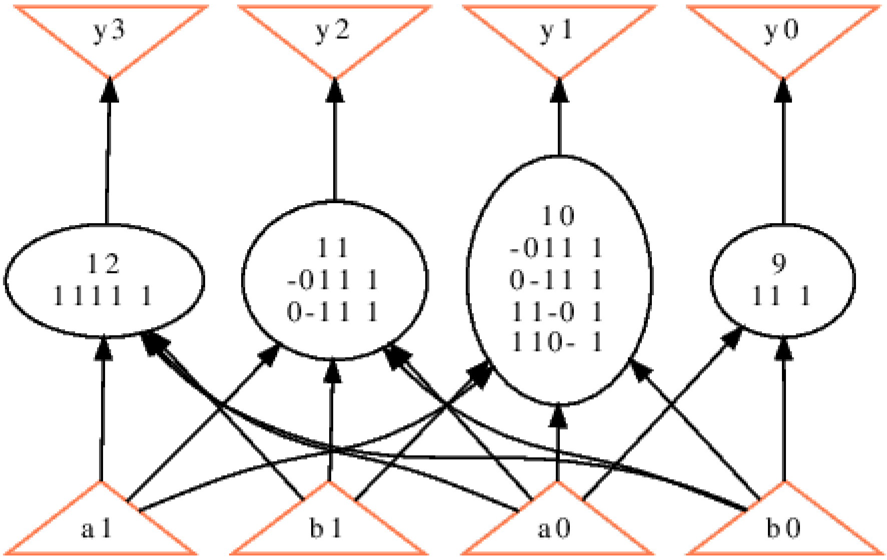
1. command “aig” vs. command “strash”

The result of the command “aig” is shown below. The result of the command “strash” is shown in problem 2.

We can see that the command “aig” only convert the local functions of the nodes to AIGs, so it do not have any difference with the original network structure from the outer look. However, for the command “strash”, it recreates the whole logic network into AIG. So, it is highly different from the original network structure.



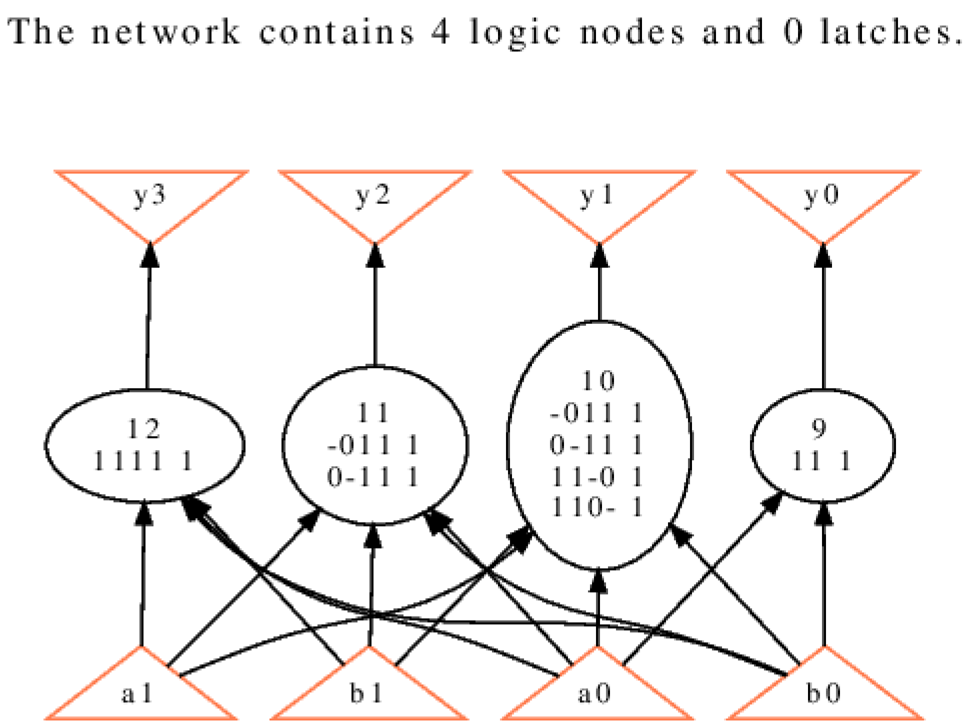
```
The network contains 4 logic nodes and 0 latches.
```



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

The result of the command “bdd” is shown below, which is shown by the command “show”. The result of the command “collapse” is shown in problem 2, which is shown by the command “show_bdd -g”.

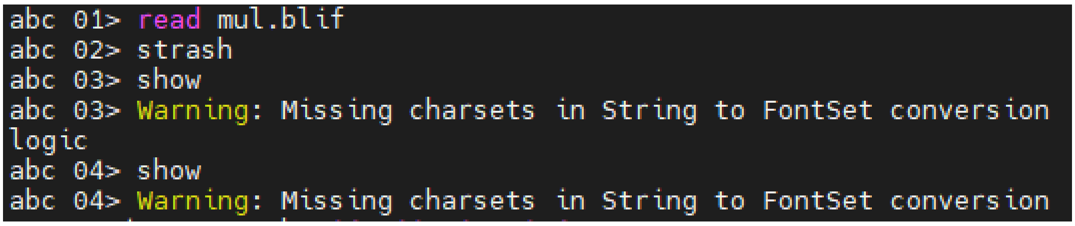
We can see that the command “bdd” only convert the local functions of the nodes to BDDs, so it do not have any difference with the original network structure from the outer look. And it is also the reason why I choose the command “show” to compare the difference with the command “collapse”. However, for the command “collapse”, it builds global functions into BDDs.



The network contains 4 logic nodes and 0 latches.

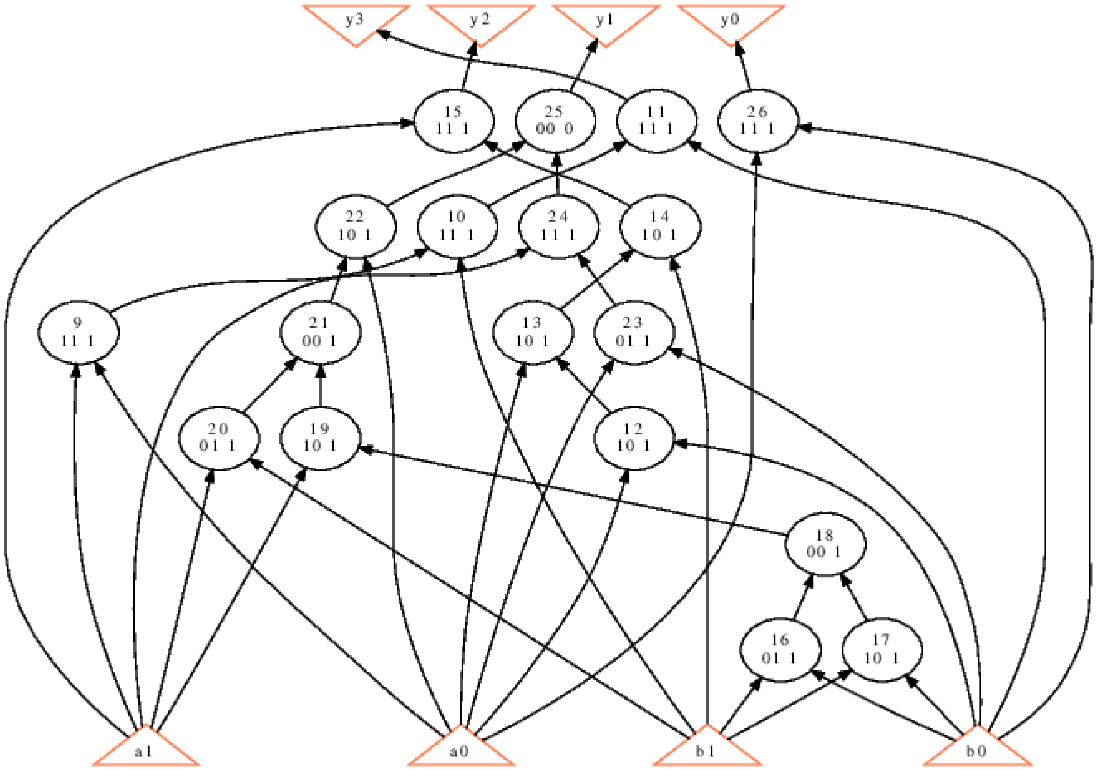
(b)

Command sequence: command “logic” then command “show”



```
abc 01> read mul.blif
abc 02> strash
abc 03> show
abc 03> Warning: Missing charsets in String to FontSet conversion
logic
abc 04> show
abc 04> Warning: Missing charsets in String to FontSet conversion
```

The result is shown below. We can see that the command “logic” transforms the AIG into a logic network with SOP representation of the two-input AND-gates, which is each nodes in the figure.


y3
y2
y1
y0
15
11 1
25
00 0
11
11 1
26
11 1
22
10 1
10
11 1
24
11 1
14
10 1
9
11 1
21
00 1
13
10 1
23
01 1
20
01 1
19
10 1
12
10 1
18
00 1
16
01 1
17
10 1
a1
a0
b1
b0