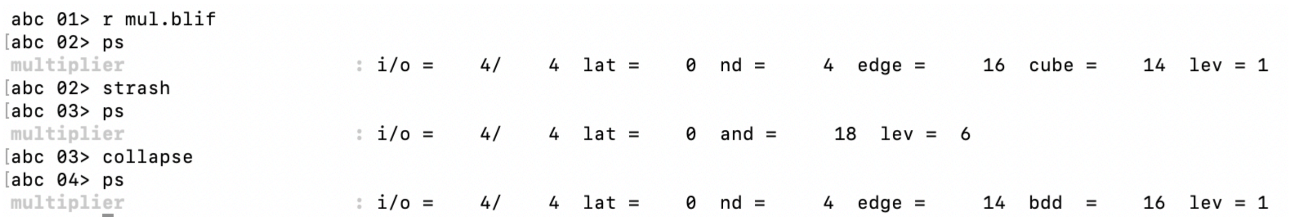
LSV PA1 Report

R10943185 葛艾

2. [Using ABC]

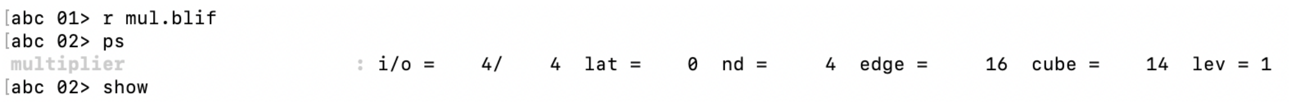
(b) Perform the following steps to practice using ABC with the two-bit unsigned multiplier example. Screenshot the results after running the commands and put them in your report.

Results of print_stats:

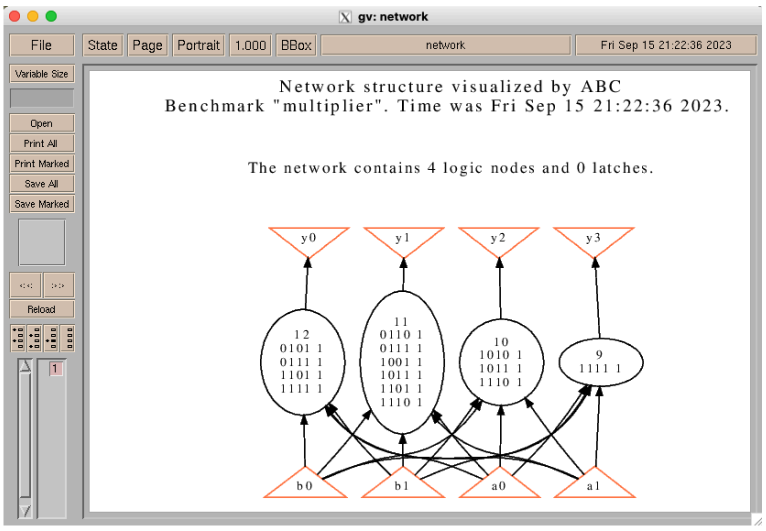


```
 abc 01> r mul.blif
[abc 02> ps
 multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     16  cube =     14  lev = 1
[abc 02> strash
[abc 03> ps
 multiplier                    : i/o =    4/    4  lat =    0  and =     18  lev =  6
[abc 03> collapse
[abc 04> ps
 multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     14  bdd  =     16  lev = 1
```

Results of "show" and "show_bdd -g" after steps 3, 5, 7:

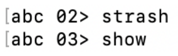


```
[abc 01> r mul.blif
[abc 02> ps
 multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     16  cube =     14  lev = 1
[abc 02> show
```

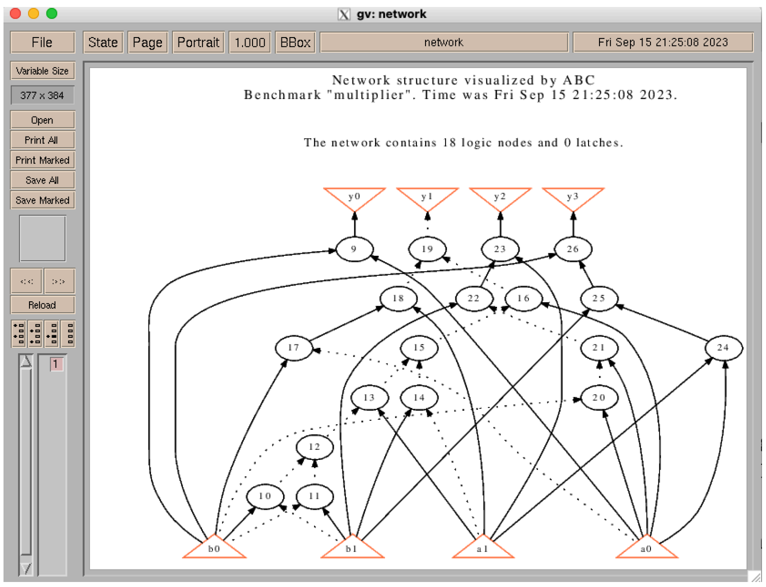


Network structure visualized by ABC
Benchmark "multiplier". Time was Fri Sep 15 21:22:36 2023.

The network contains 4 logic nodes and 0 latches.



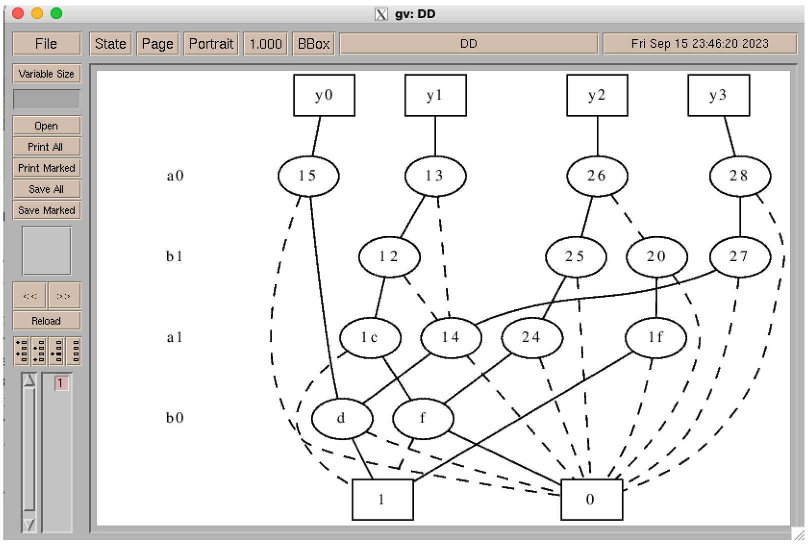
```
[abc 02> strash
[abc 03> show
```



Network structure visualized by ABC
Benchmark "multiplier". Time was Fri Sep 15 21:25:08 2023.

The network contains 18 logic nodes and 0 latches.

```
[abc 03> collapse
[abc 04> show_bdd -g
```



3. [ABC Boolean Function Representations]
(a) Compare the following differences with the two-bit unsigned multiplier example.

1. logic network in AIG (command “aig”) vs. structurally hashed AIG (command “strash”)

   The difference between command “aig" and "strash" lies in the following aspects. Command “aig" transforms individual (local) nodes into AIGs while preserving the original network's structure. Conversely, "strash" collapses the entire network into a single, highly-reduced AIG. As a result, the network after applying "strash" becomes simpler and smaller.

```
[abc 01> r mul.blif
[abc 02> aig
[abc 02> ps
multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     16  aig  =    18  lev = 1
[abc 02> strash
[abc 03> ps
multiplier                    : i/o =    4/    4  lat =    0  and =    18  lev =  6
```

2. logic network in BDD (command “bdd”) vs. collapsed BDD (command “collapse”)

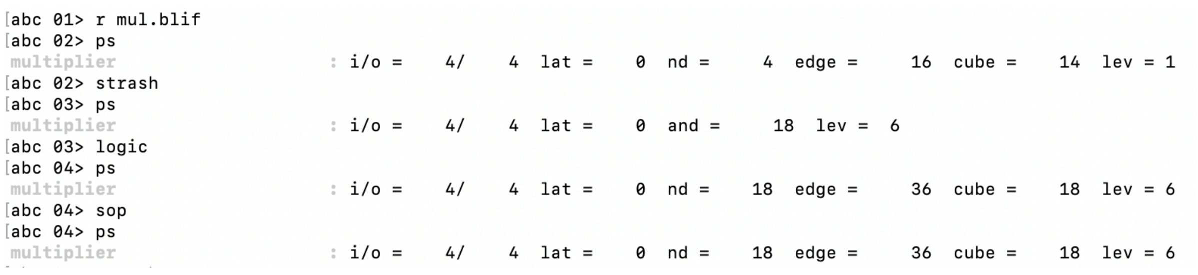
   The difference between command “bdd” and “collapse” lies in the following aspects. “bdd” converts the individual (local) nodes into BDDs, while preserving the original network's structure. On the contrary, collapses the entire network into a single, highly-reduced BDD, reducing the graph size.

```
[abc 01> r mul.blif
[abc 02> bdd
[abc 02> ps
multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     16  bdd  =    17  lev = 1
[abc 02> collapse
[abc 03> ps
multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     14  bdd  =    16  lev = 1
```

(b) Given a structurally hashed AIG, find a sequence of ABC commands to convert it to a logic network with node function expressed in sum-of-products (SOP). Use the two-bit unsigned multiplier example to test your command sequence, screenshot the results, and put them in your report.

1. Read design

2. Command "logic": Convert the AIG into a logic network (SOP representation)



```
abc 01> r mul.blif
abc 02> ps
multiplier                    : i/o =    4/    4  lat =    0  nd =     4  edge =     16  cube =    14  lev = 1
abc 02> strash
abc 03> ps
multiplier                    : i/o =    4/    4  lat =    0  and =     18  lev =  6
abc 03> logic
abc 04> ps
multiplier                    : i/o =    4/    4  lat =    0  nd =    18  edge =     36  cube =    18  lev = 6
abc 04> sop
abc 04> ps
multiplier                    : i/o =    4/    4  lat =    0  nd =    18  edge =     36  cube =    18  lev = 6
```