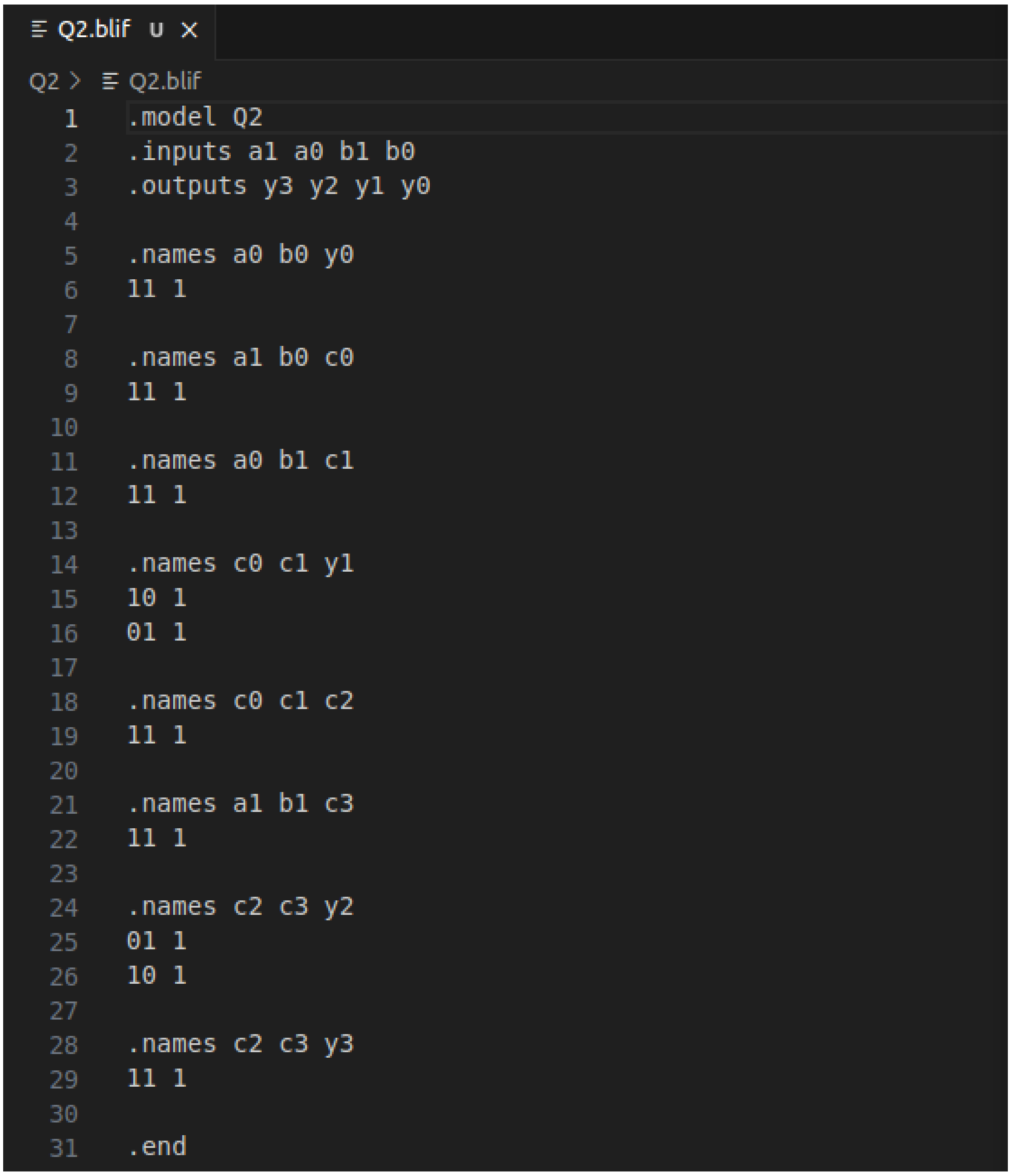
Logic Synthesis & Verification

Programming Assignment 1

2.

(a).



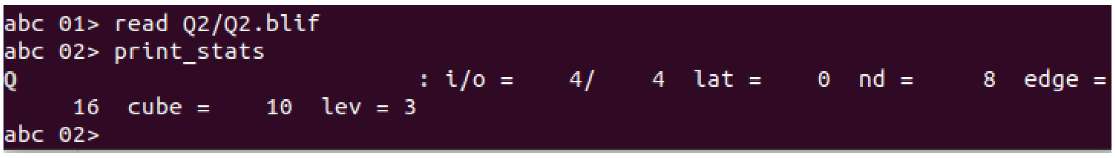
Q2.blif U ×

Q2 > Q2.blif

```
.model Q2
.inputs a1 a0 b1 b0
.outputs y3 y2 y1 y0

.names a0 b0 y0
11 1

.names a1 b0 c0
11 1

.names a0 b1 c1
11 1

.names c0 c1 y1
10 1
01 1

.names c0 c1 c2
11 1

.names a1 b1 c3
11 1

.names c2 c3 y2
01 1
10 1

.names c2 c3 y3
11 1

.end
```

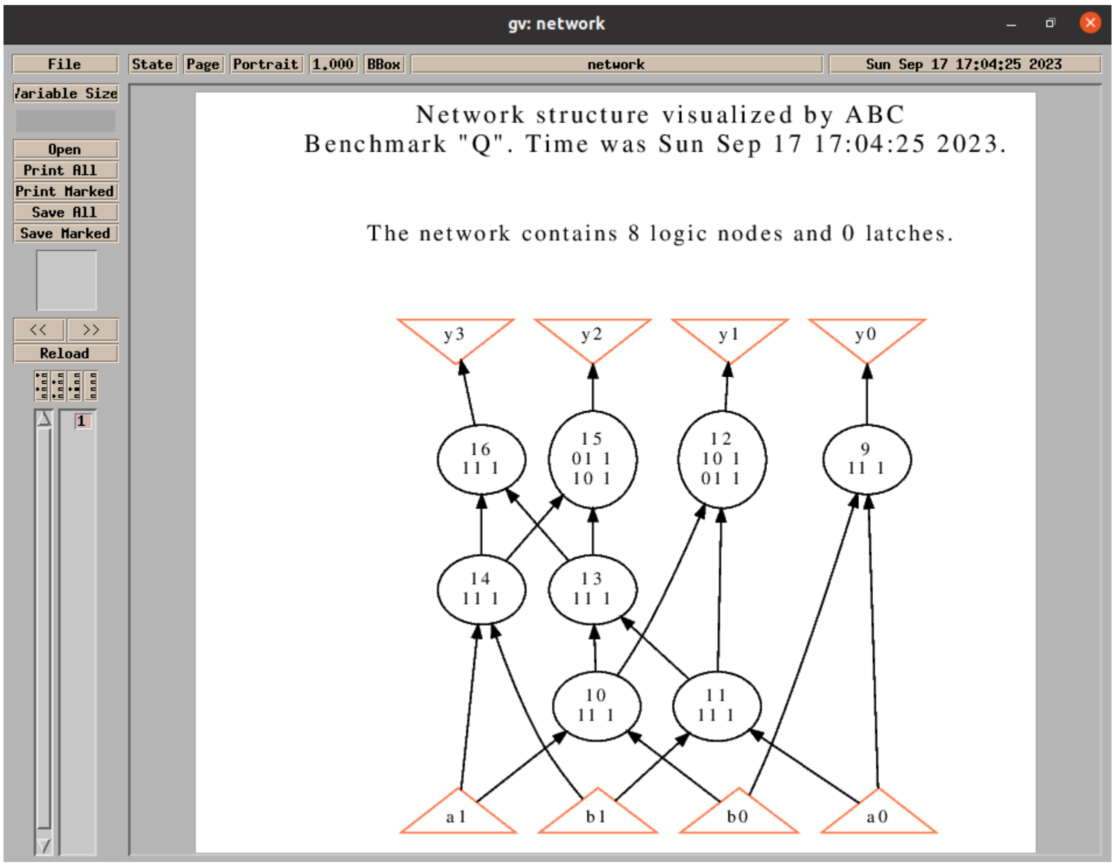
(b)

read then ***print_stats***:



```
abc 01> read Q2/Q2.blif
abc 02> print_stats
Q                              : i/o =    4/    4  lat =    0  nd =     8  edge =
     16  cube =    10  lev = 3
abc 02>
```

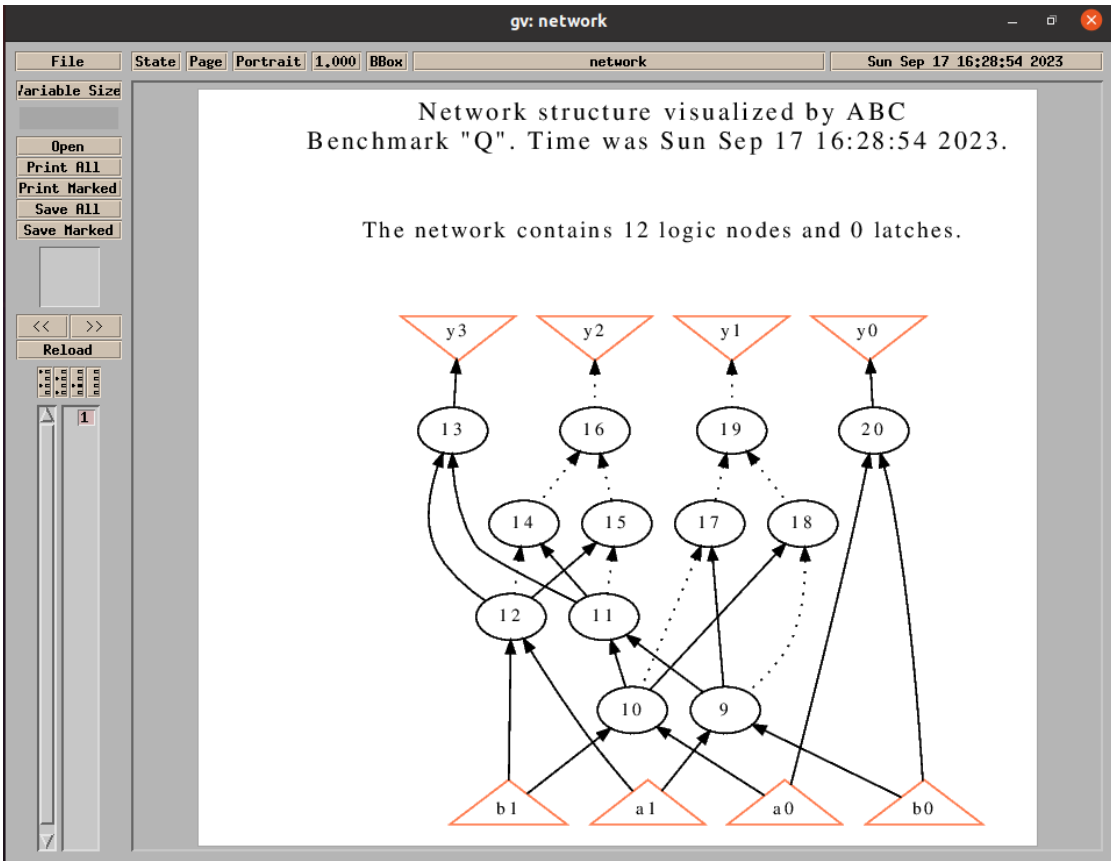
show:



Network structure visualized by ABC
Benchmark "Q". Time was Sun Sep 17 17:04:25 2023.

The network contains 8 logic nodes and 0 latches.

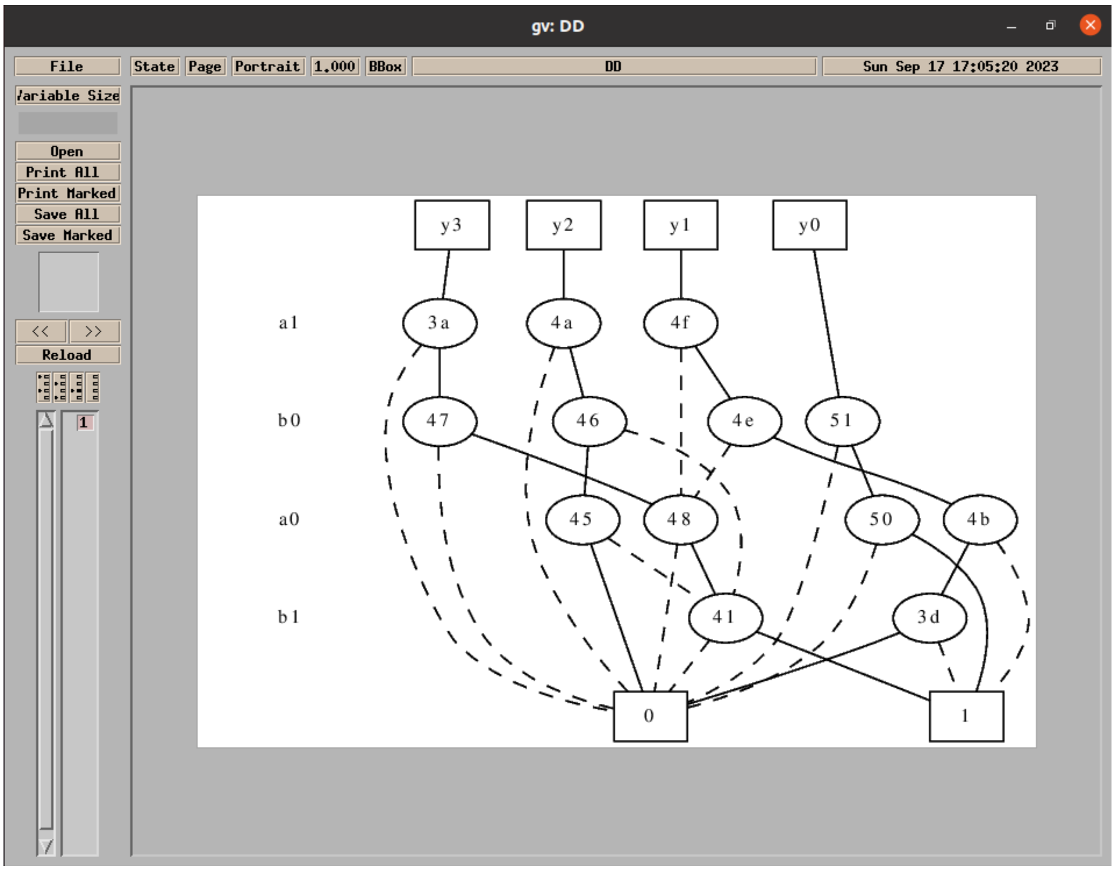
strash then ***show***:



Network structure visualized by ABC
Benchmark "Q". Time was Sun Sep 17 16:28:54 2023.

The network contains 12 logic nodes and 0 latches.

collapse then ***show_bdd***:

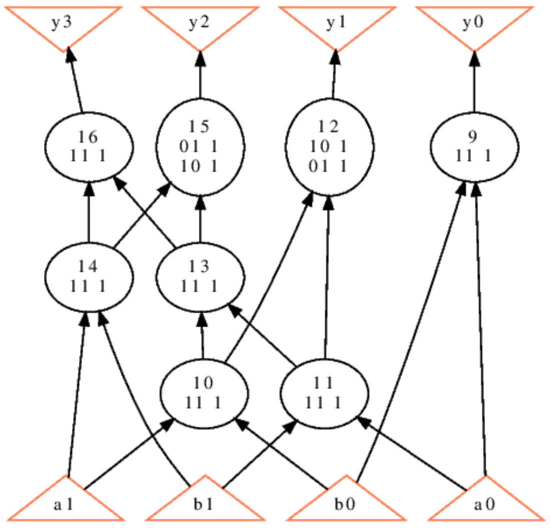


3.

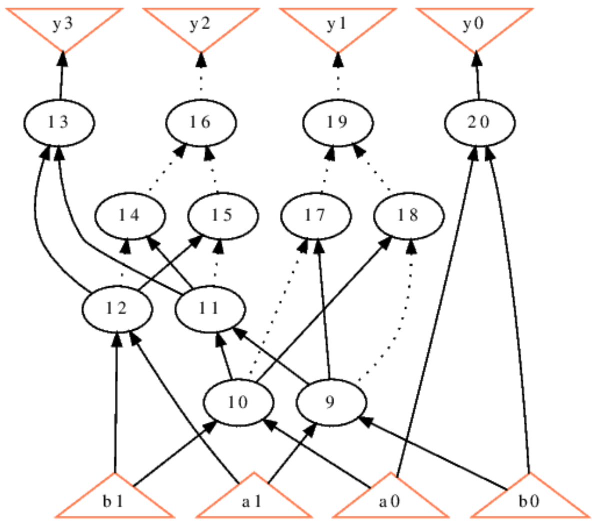
(a).

1.

aig:

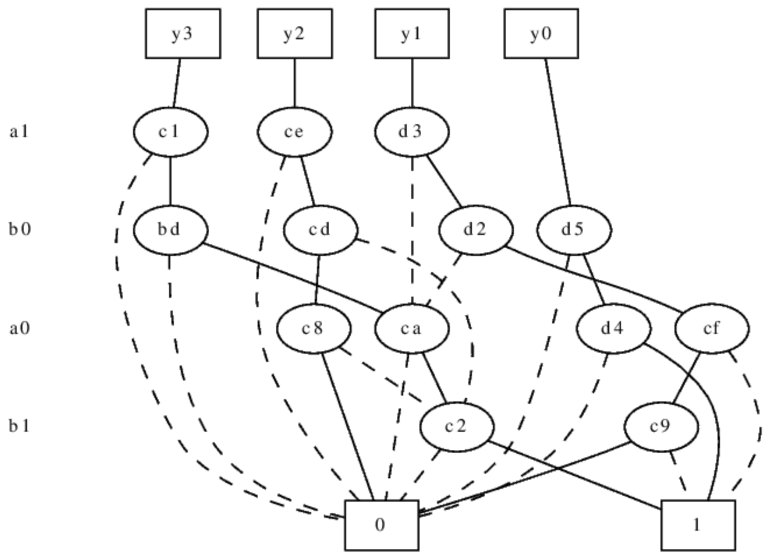


strash:

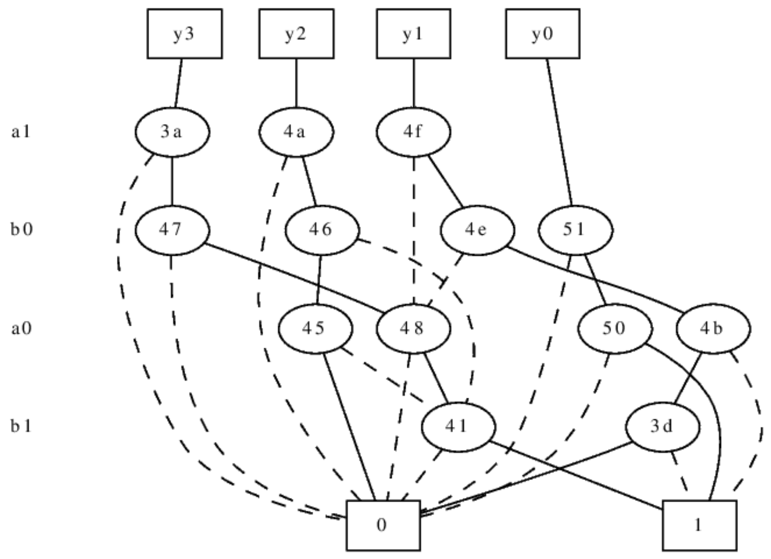


Differences: each node in ***aig*** forms a boolean function, while ***strash*** converts the network into And-Inverter graph.

2.***bdd***

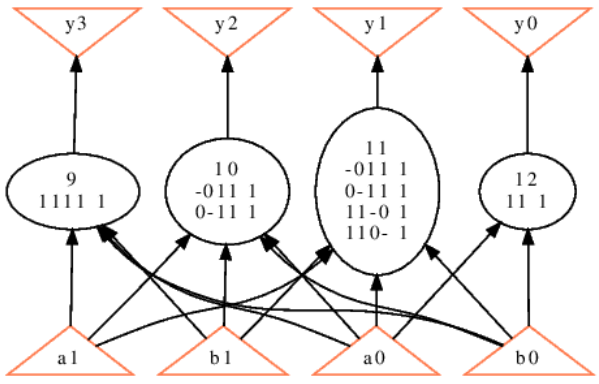


collapse:

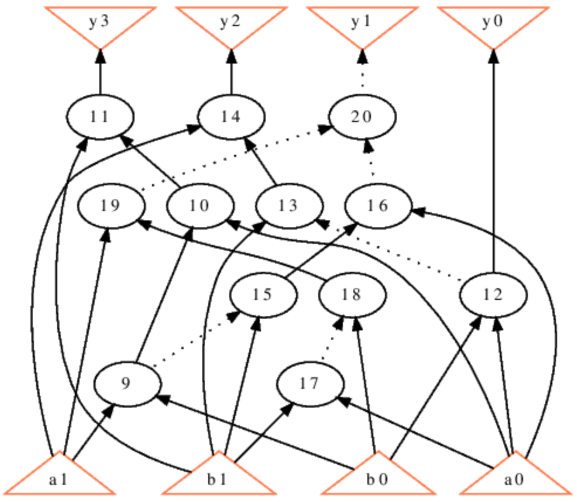


difference: ***bdd*** converts the local functions of the nodes to BDDs, while ***collapse*** builds the global functions with BDDs.

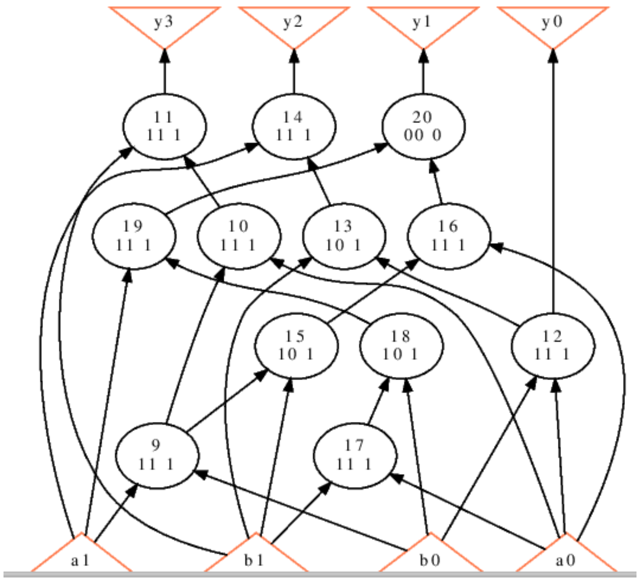
(b).For a structurally hashed AIG, use command ***strash*** then ***logic*** to get the result.
original(use structurally hashed AIG generated in prob.2-(b)-6 as example):



After ***strash***:



After ***logic***:


y 3
y 2
y 1
y 0
1 1
11 1
1 4
11 1
20
00 0
19
11 1
1 0
11 1
13
10 1
1 6
11 1
1 5
10 1
1 8
10 1
1 2
11 1
9
11 1
1 7
11 1
a 1
b 1
b 0
a 0