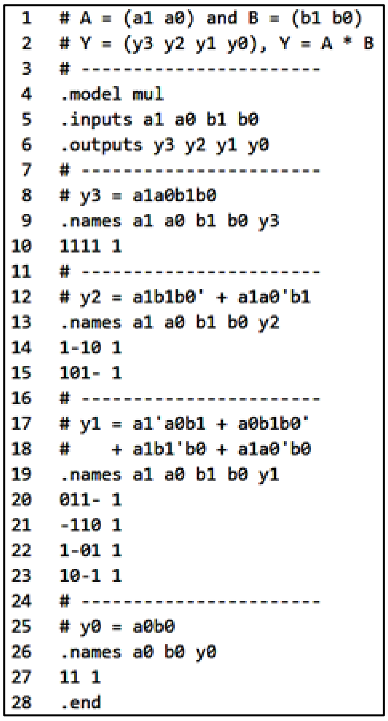
Submit Exercises 1 and 4 on GitHub, and **Exercises 2 and 3** on NTU Cool.

Exercise 2 (a) [Using ABC]



```
# A = (a1 a0) and B = (b1 b0)
# Y = (y3 y2 y1 y0), Y = A * B
# -----------------------
.model mul
.inputs a1 a0 b1 b0
.outputs y3 y2 y1 y0
# -----------------------
# y3 = a1a0b1b0
.names a1 a0 b1 b0 y3
1111 1
# -----------------------
# y2 = a1b1b0' + a1a0'b1
.names a1 a0 b1 b0 y2
1-10 1
101- 1
# -----------------------
# y1 = a1'a0b1 + a0b1b0'
#     + a1b1'b0 + a1a0'b0
.names a1 a0 b1 b0 y1
011- 1
-110 1
1-01 1
10-1 1
# -----------------------
# y0 = a0b0
.names a0 b0 y0
11 1
.end
```

mul.blif

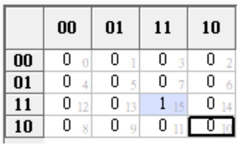
A = (a_1 a_0), B = (b_1 b_0)

Y = (y_3 y_2 y_1 y_0), Y = A * B

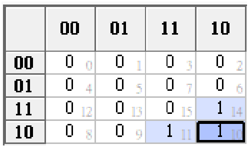
Construct the truth table in Table 1 and draw Karnaugh maps to get Boolean equations:

$y_3 = a_1a_0b_1b_0$

$y_2 = a_1b_1b_0' + a_1a_0'b_1$



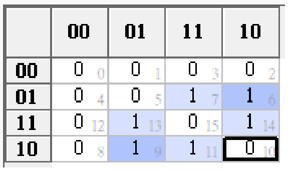
	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	0 5	0 7	0 6
11	0 12	0 13	1 15	0 14
10	0 8	0 9	0 11	0 10



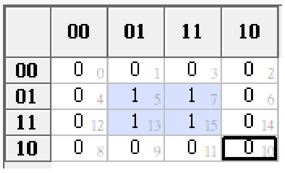
	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	0 5	0 7	0 6
11	0 12	0 13	0 15	1 14
10	0 8	0 9	1 11	1 10

$y_1 = a_1'a_0b_1 + a_0b_1b_0' + a_1b_1'b_0 + a_1a_0'b_0$

$y_0 = a_0b_0$



	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	0 5	1 7	1 6
11	0 12	1 13	0 15	1 14
10	0 8	1 9	1 11	0 10



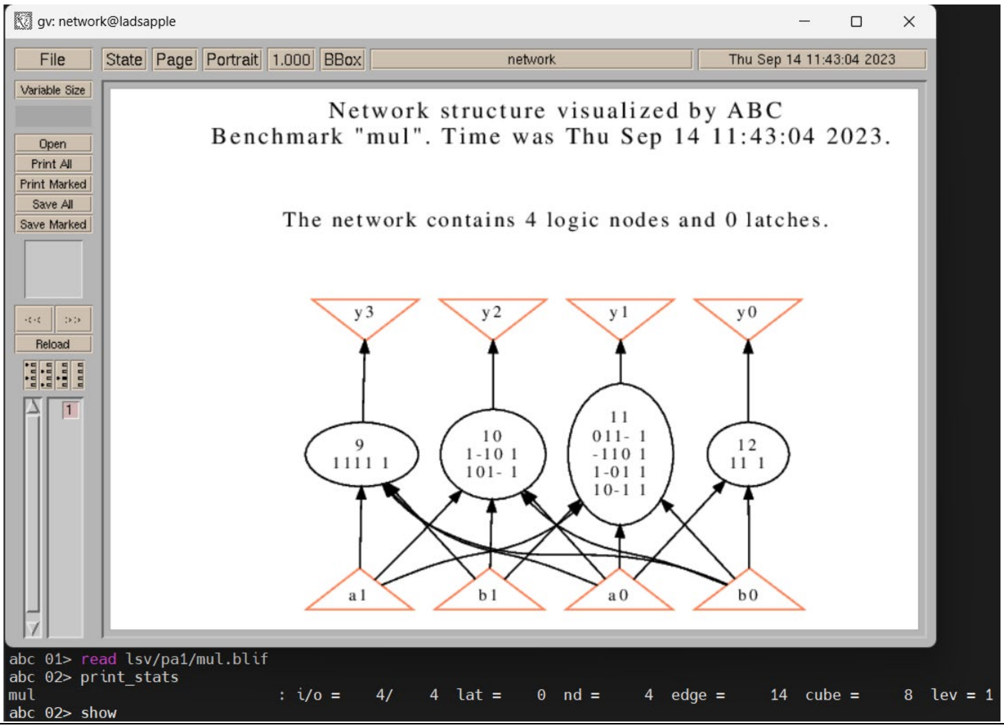
	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	1 5	1 7	0 6
11	0 12	1 13	1 15	0 14
10	0 8	0 9	0 11	0 10

Table 1. Truth table of the four-bit unsigned integer in Exercise 2

a_1	a_0	b_1	b_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Exercise 2 (b) [Using ABC]

1. read the BLIF file into ABC (command “read”)
2. check statistics (command “print stats”)
3. visualize the network structure (command “show”)

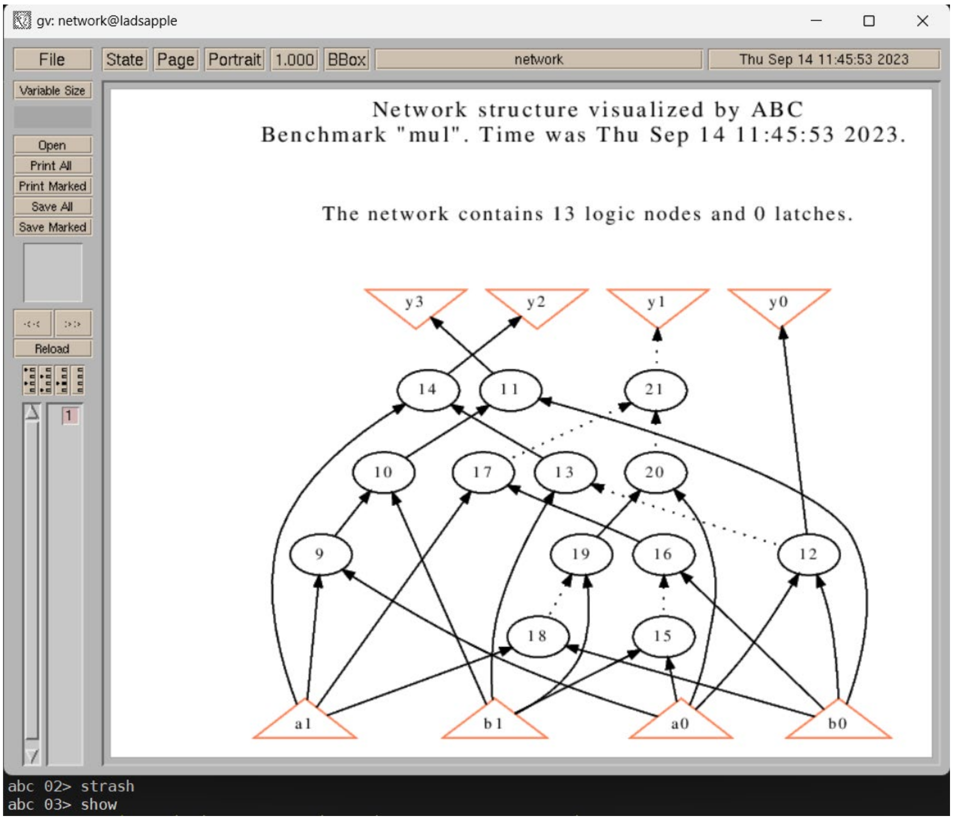


Network structure visualized by ABC
Benchmark "mul". Time was Thu Sep 14 11:43:04 2023.

The network contains 4 logic nodes and 0 latches.

```
abc 01> read lsv/pa1/mul.blif
abc 02> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =    4  edge =     14  cube =     8  lev = 1
abc 02> show
```

4. convert to AIG (command “strash”)
5. visualize the AIG (command “show”)



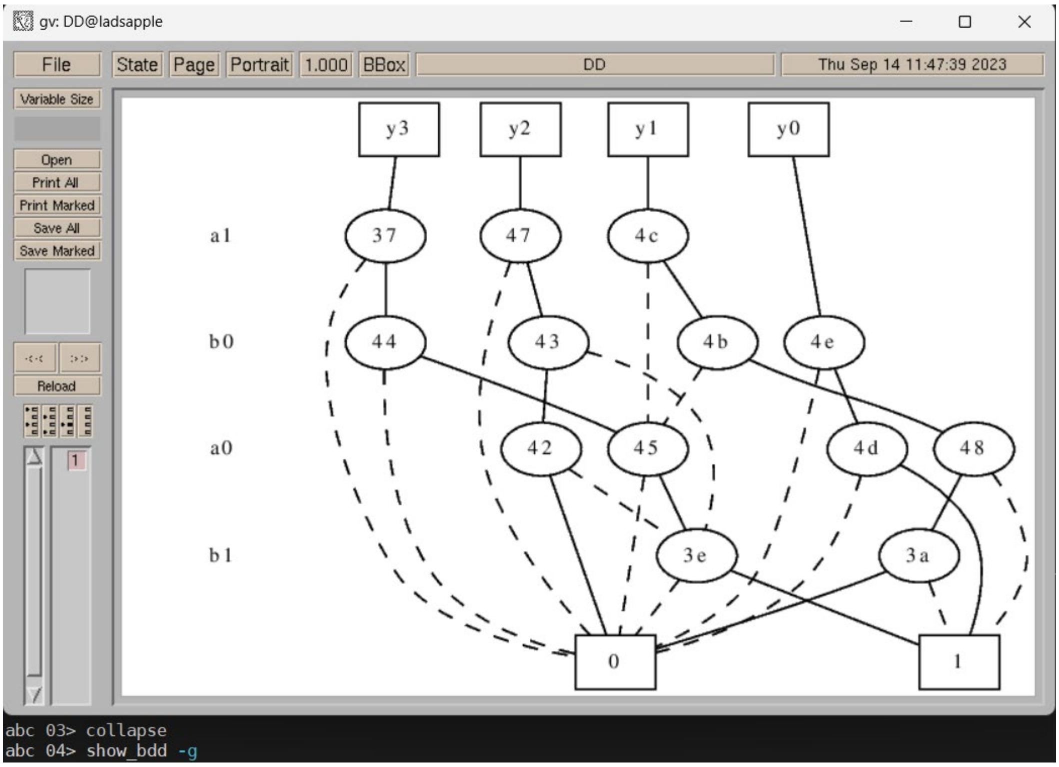
Network structure visualized by ABC
Benchmark "mul". Time was Thu Sep 14 11:45:53 2023.

The network contains 13 logic nodes and 0 latches.

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

6. convert to BDD (command “collapse”)

7. visualize the BDD (command “show bdd -g”; note that “show bdd” only shows the first PO; option “-g” can be applied to show all POs)



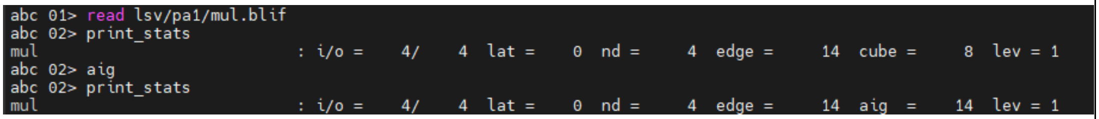
```
abc 03> collapse
abc 04> show_bdd -g
```

Exercise 3 (a) [ABC Boolean Function Representations]]

1. **logic network in AIG** vs. **structurally hashed AIG**

(1) **logic network in AIG** (by command “aig”)

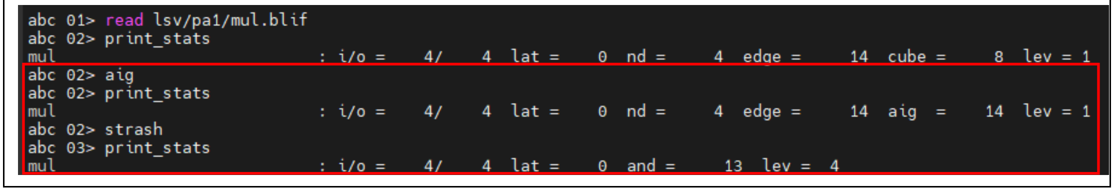
- **aig** converts local functions of the nodes to AIGs.
- print_stats shows that 8 cubes become 14 aigs.



```
abc 01> read lsv/pa1/mul.blif
abc 02> print_stats
mul                          : i/o =    4/    4  lat =    0  nd =     4  edge =     14  cube =     8  lev = 1
abc 02> aig
abc 02> print_stats
mul                          : i/o =    4/    4  lat =    0  nd =     4  edge =     14  aig  =    14  lev = 1
```

(2) structurally hashed AIG (by command “strash”)

- **strash** transforms the current network into an AIG by one-level structural hashing. The resulting AIG is a logic network composed of two-input AND gates and inverters represented as complemented attributes on the edges.
- logic network in AIG vs. structurally hashed AIG:
  4 nds, 14 edges, 14 aigs, 1 lev vs. 13 ands, 4 levs

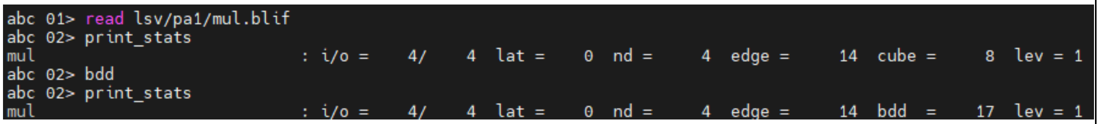


```
abc 01> read lsv/pa1/mul.blif
abc 02> print_stats
mul                          : i/o =    4/    4  lat =    0  nd =     4  edge =     14  cube =     8  lev = 1
abc 02> aig
abc 02> print_stats
mul                          : i/o =    4/    4  lat =    0  nd =     4  edge =     14  aig  =    14  lev = 1
abc 02> strash
abc 03> print_stats
mul                          : i/o =    4/    4  lat =    0  and =     13  lev =  4
```

2. **logic network in BDD** vs. **collapsed BDD**

(1) logic network in BDD (by command "bdd")

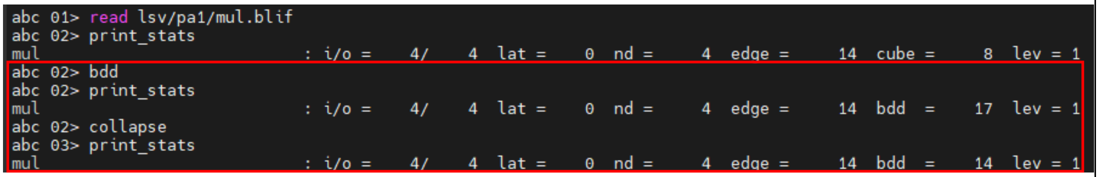
- ➢ Converts local functions of the nodes to BDDs.
- ➢ print_stats shows that 8 cubes become 17 bdds.



```
abc 01> read lsv/pa1/mul.blif
abc 02> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =     4  edge =     14  cube =     8  lev = 1
abc 02> bdd
abc 02> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =     4  edge =     14  bdd  =    17  lev = 1
```

(2) collapsed BDD (by command "collapse")

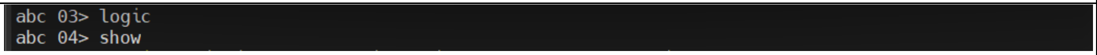
- ➢ Recursively composes the fanin nodes into the fanout nodes resulting in a network, in which each CO is produced by a node, whose fanins are CIs. Collapsing is performed by building global functions using BDDs and is, therefore, limited to relatively small circuits. After collapsing, the node functions are represented using BDDs.
- ➢ logic network in BDD vs. collapsed BDD: 17 bdds vs. 14 bdds.



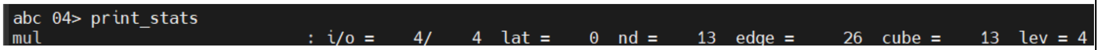
```
abc 01> read lsv/pa1/mul.blif
abc 02> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =     4  edge =     14  cube =     8  lev = 1
abc 02> bdd
abc 02> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =     4  edge =     14  bdd  =    17  lev = 1
abc 02> collapse
abc 03> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =     4  edge =     14  bdd  =    14  lev = 1
```

Exercise 3 (a) [ABC Boolean Function Representations]]

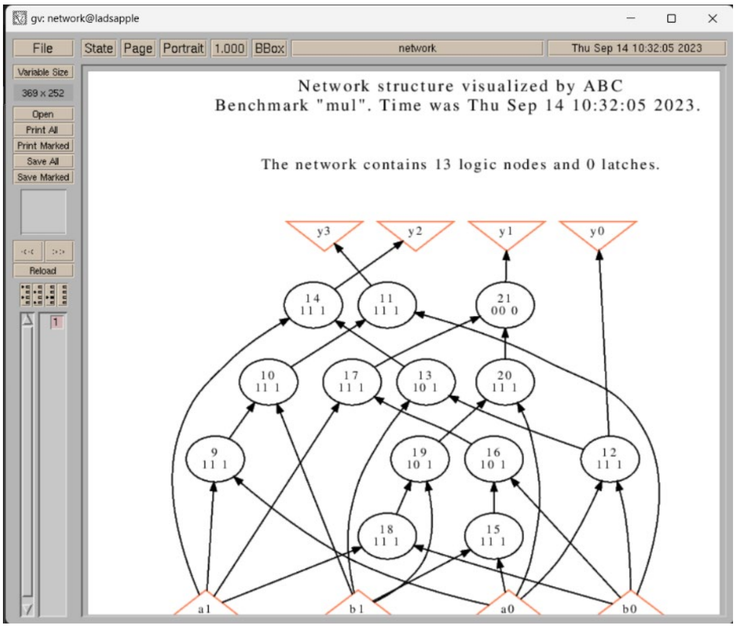
- ➢ Simply type command "**logic**"
- ➢ **logic** transforms the AIG into a logic network with the SOP representation of the two-input AND-gates.



```
abc 03> logic
abc 04> show
```



```
abc 04> print_stats
mul                           : i/o =    4/    4  lat =    0  nd =    13  edge =     26  cube =    13  lev = 4
```



Network structure visualized by ABC
Benchmark "mul". Time was Thu Sep 14 10:32:05 2023.

The network contains 13 logic nodes and 0 latches.