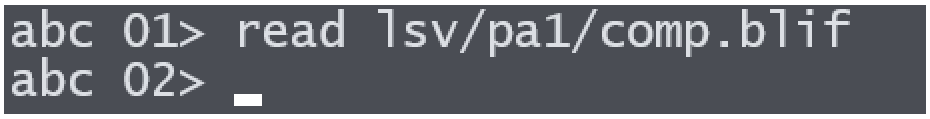
Logic Synthesis & Verification, Fall 2024

Report for Exercise 2, 3 of Programming Assignment 1

B10901022 Shih-En Chou (周世恩)

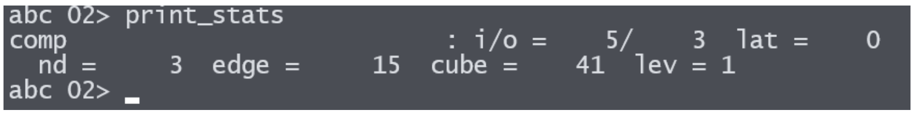
2(b)

1.



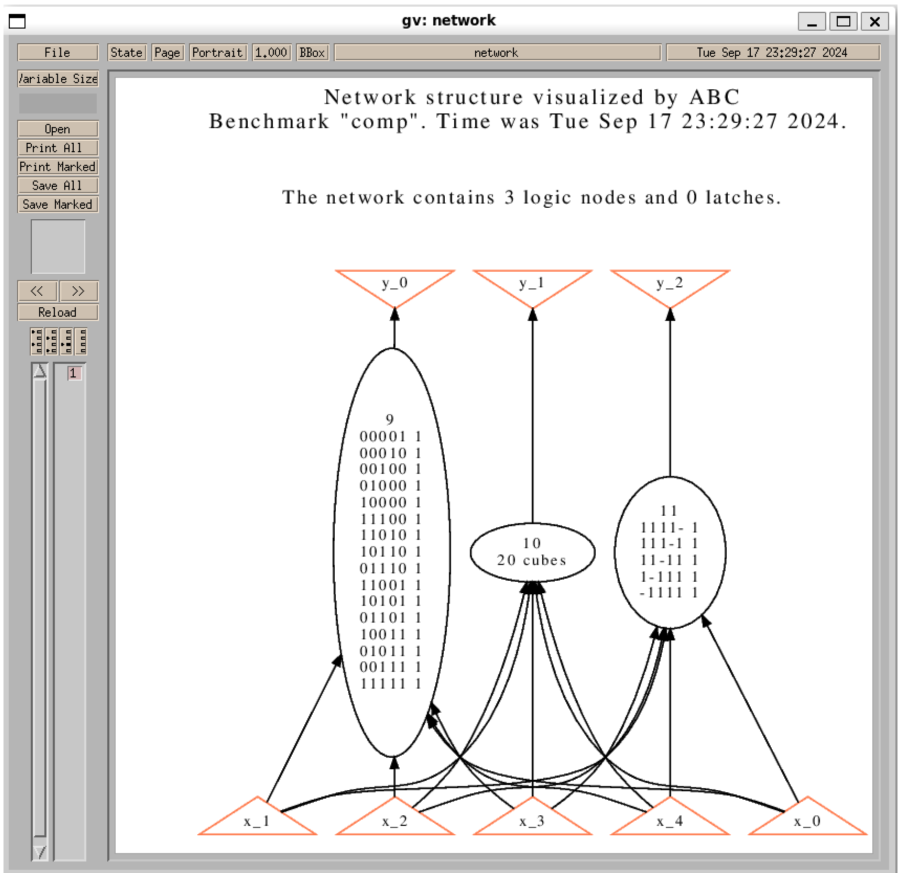
```
abc 01> read lsv/pa1/comp.blif
abc 02> _
```

2.



```
abc 02> print_stats
comp                              : i/o =    5/    3  lat =    0
  nd =     3  edge =     15  cube =    41  lev = 1
abc 02> _
```

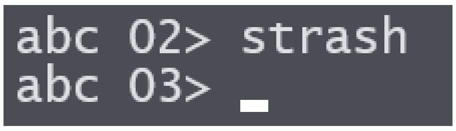
3.



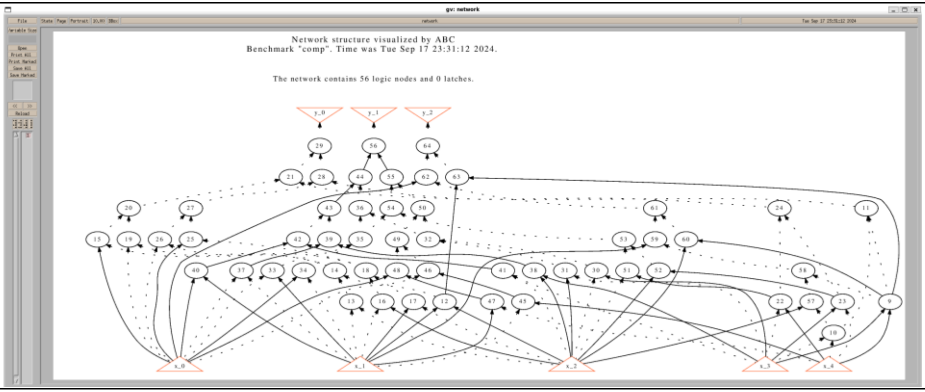
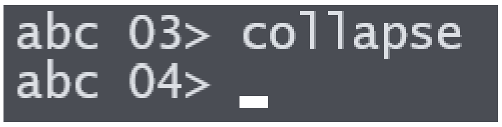
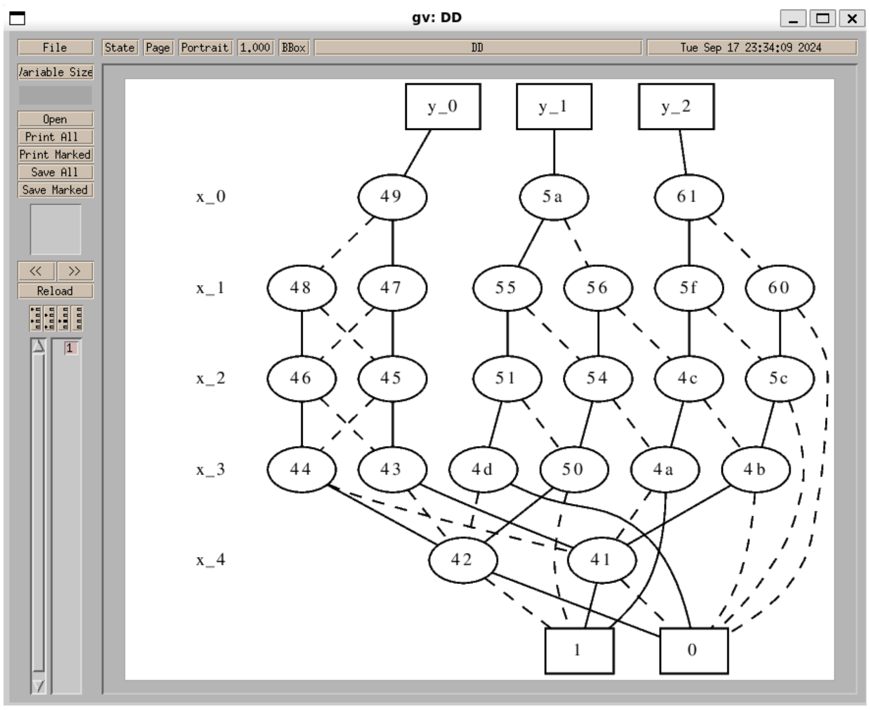
Network structure visualized by ABC
Benchmark "comp". Time was Tue Sep 17 23:29:27 2024.

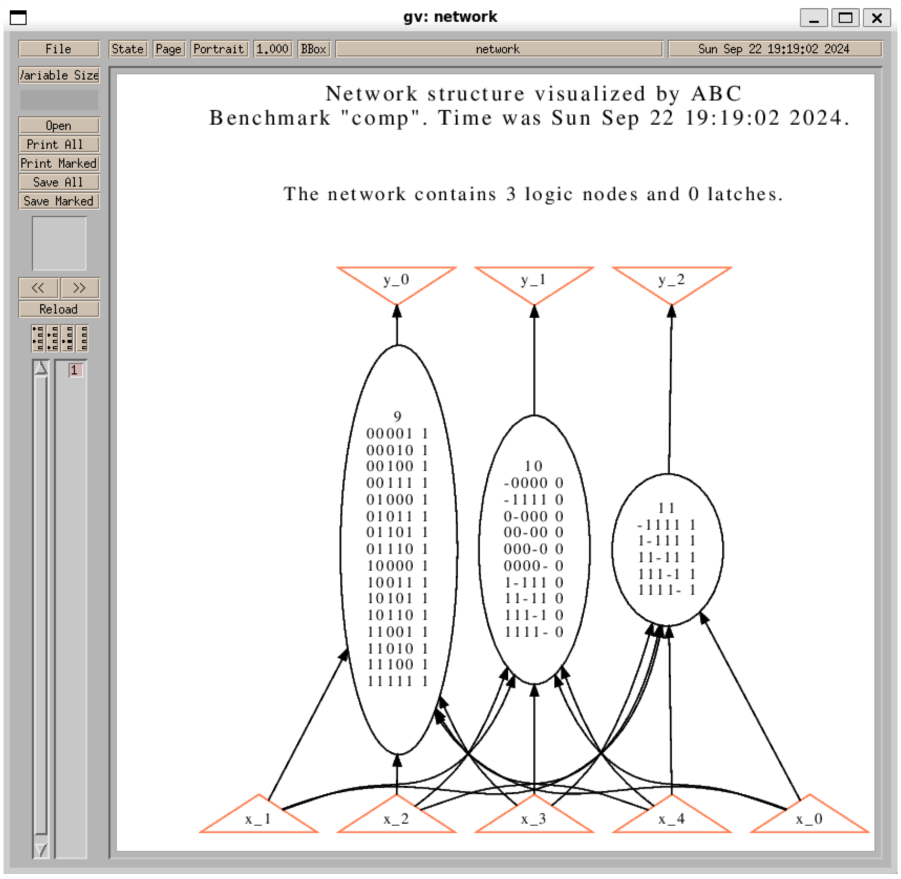
The network contains 3 logic nodes and 0 latches.

4.



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

	5.	 Network structure visualized by ABC Benchmark "comp". Time was Tue Sep 17 23:31:12 2024. The network contains 56 logic nodes and 0 latches.
	6.	 `abc 03> collapse` `abc 04> _`
	7.	 gv: DD File State Page Portrait 1.000 BBox DD Tue Sep 17 23:34:09 2024 Variable Size Open Print All Print Marked Save All Save Marked Reload y_0 y_1 y_2 x_0 x_1 x_2 x_3 x_4 1 0
3(a)	1.	After typing the command *aig*, typing *show* gives



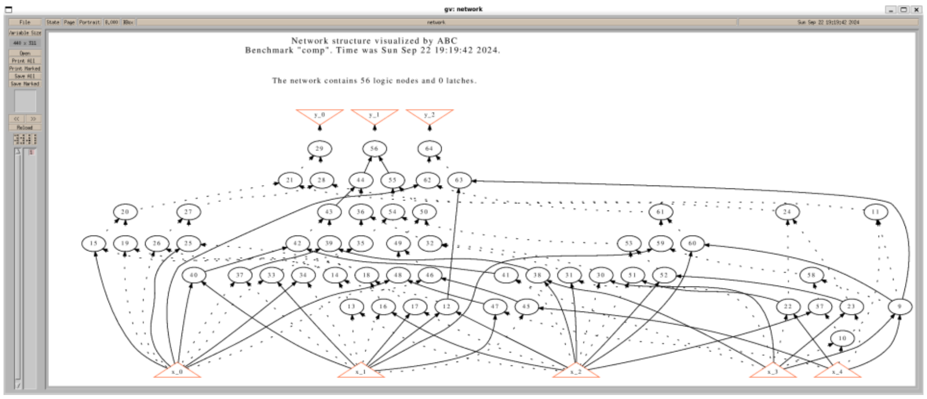
Network structure visualized by ABC
Benchmark "comp". Time was Sun Sep 22 19:19:02 2024.

The network contains 3 logic nodes and 0 latches.

and *print_stats* gives

```
comp                          : i/o =     5/    3  lat =    0  nd =     3  edge =
15  aig  =    61  lev = 1
```

while after using *strash* the two commands give



Network structure visualized by ABC
Benchmark "comp". Time was Sun Sep 22 19:19:42 2024.

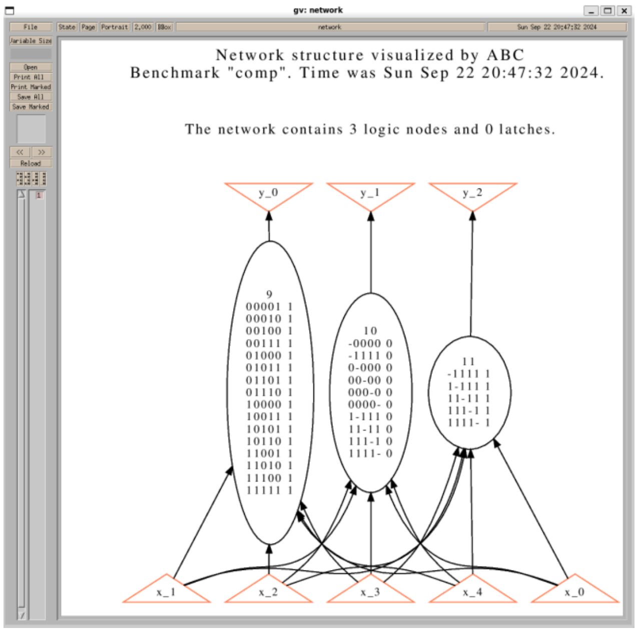
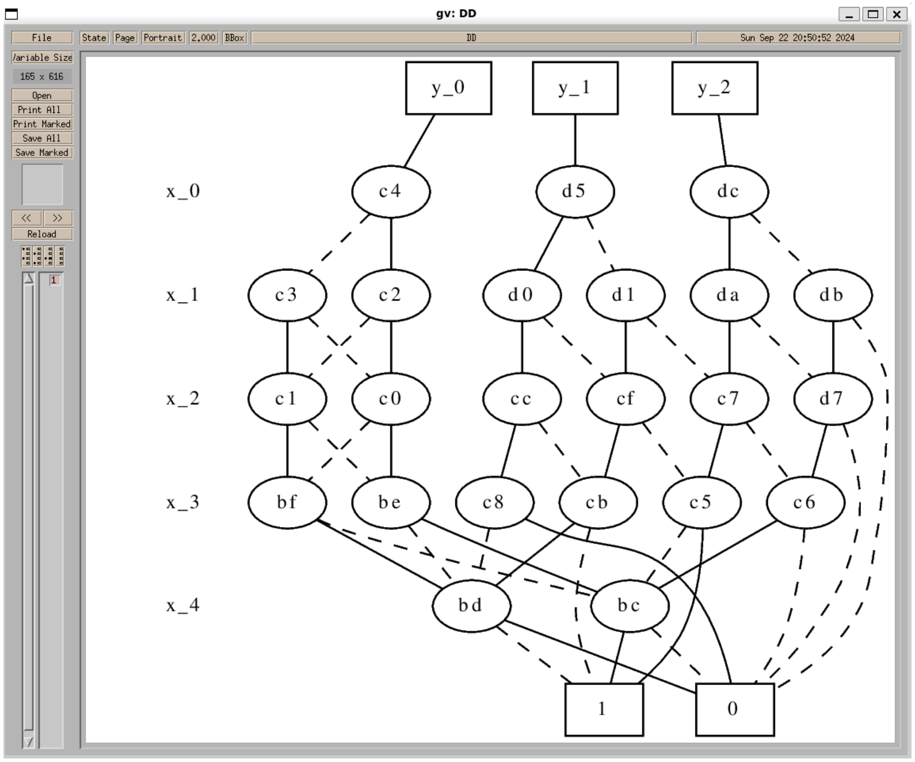
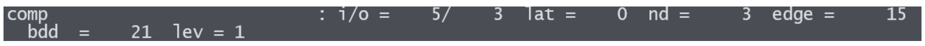
The network contains 56 logic nodes and 0 latches.

and

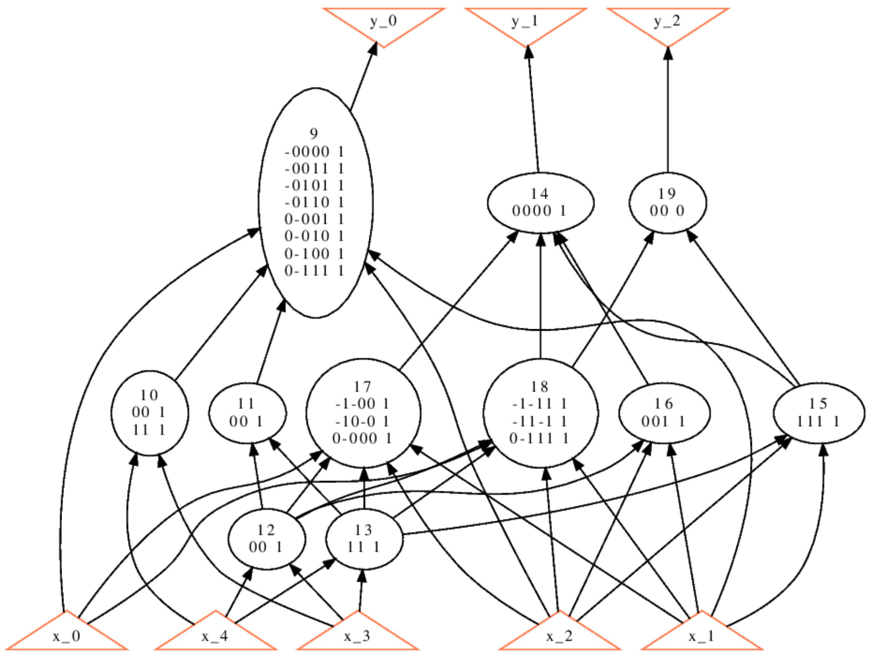
```
comp                          : i/o =     5/    3  lat =    0  and =     56  lev =  7
```

The reason for this is that *aig* only modifies how the local function of nodes are stored, while *strash* transforms the whole network into an AIG. We can observe that the network has the same number of nodes and

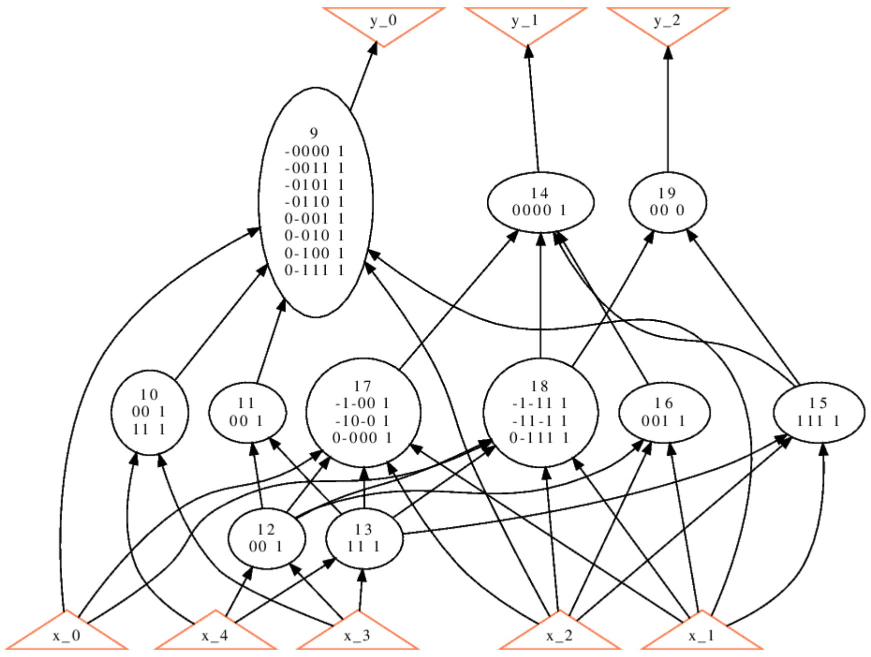
		edges even after using *aig*, but not *strash*.

	2.	*show* and *print_stats* gives the same output after typing either *bdd* or *collapse*:
		show gives
		
		gv: network
		Network structure visualized by ABC
		Benchmark "comp". Time was Sun Sep 22 20:47:32 2024.
		The network contains 3 logic nodes and 0 latches.
		`9`
		`00001 1`
		`00010 1`
		`00100 1`
		`00111 1`
		`01000 1`
		`01011 1`
		`01101 1`
		`01110 1`
		`10000 1`
		`10011 1`
		`10101 1`
		`10110 1`
		`11001 1`
		`11010 1`
		`11100 1`
		`11111 1`
		`10`
		`-0000 0`
		`-1111 0`
		`0-000 0`
		`00-00 0`
		`000-0 0`
		`0000- 0`
		`1-111 0`
		`11-11 0`
		`111-1 0`
		`1111- 0`
		`11`
		`-1111 1`
		`1-111 1`
		`11-11 1`
		`111-1 1`
		`1111- 1`
		
		, *show_bdd -g* gives
		
		 gv: DD
		y_0
		y_1
		y_2
		x_0
		x_1
		x_2
		x_3
		x_4
		c4
		d5
		dc
		c3
		c2
		d0
		d1
		da
		db
		c1
		c0
		cc
		cf
		c7
		d7
		bf
		be
		c8
		cb
		c5
		c6
		bd
		bc
		1
		0
		(the node name might be different, but it's the same bdd)
		and *print_stats* gives
		
		`comp                          : i/o =    5/    3  lat =    0  nd =     3  edge =     15`
		`  bdd  =     21  lev = 1`

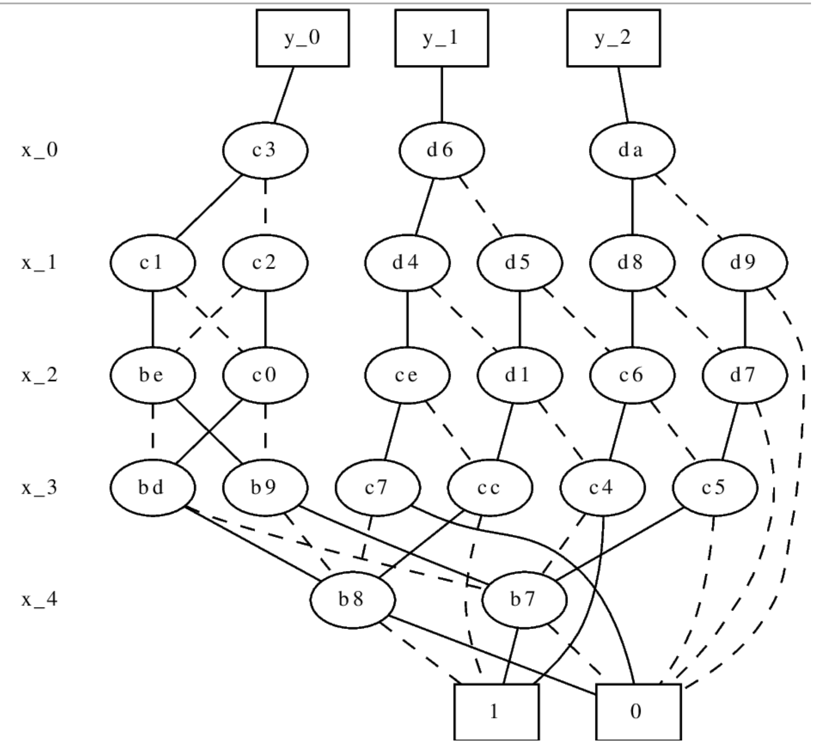
At first glance, it might be hard to tell how these two commands are different. However, this is due to the fact that the logic network we used has only one level. If we take an equivalent multi-level network



After *bdd*
show



show_bdd -g

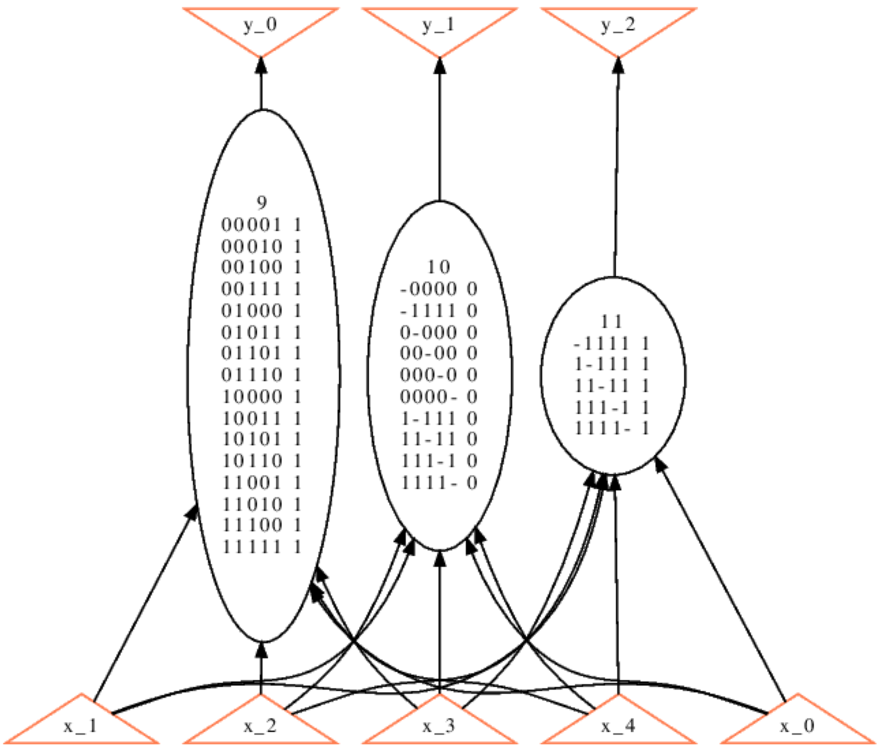


print_stats

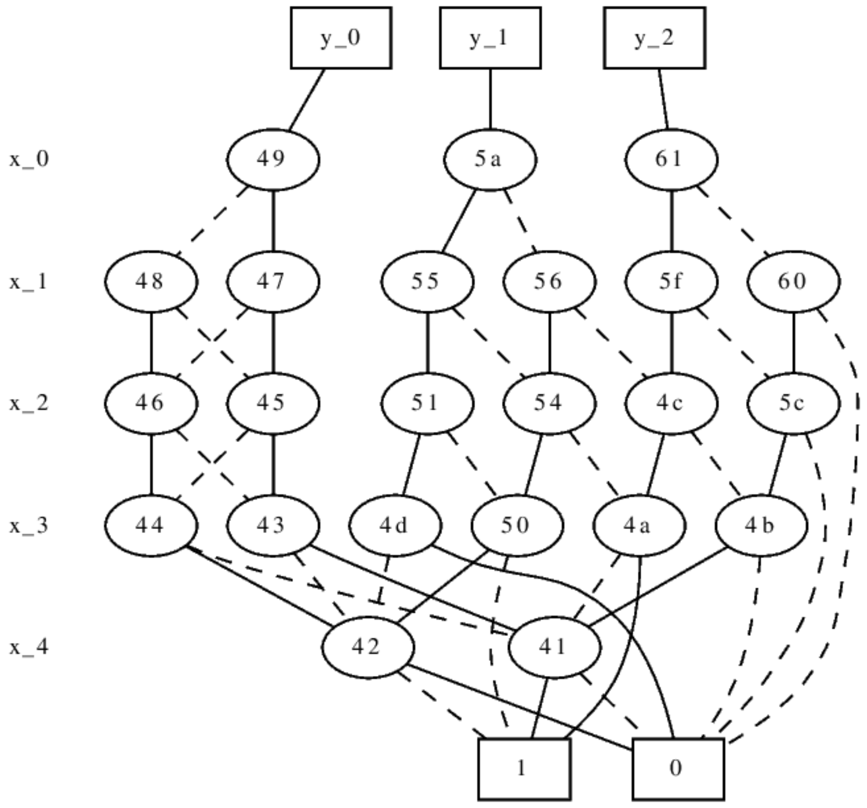
```
comp                          : i/o =    5/    3  lat =    0  nd =    11  edge =    35
  bdd  =    42  lev = 3
```

After *collapse*

show



show_bdd -g



(essentially the same as above)

print_stats

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

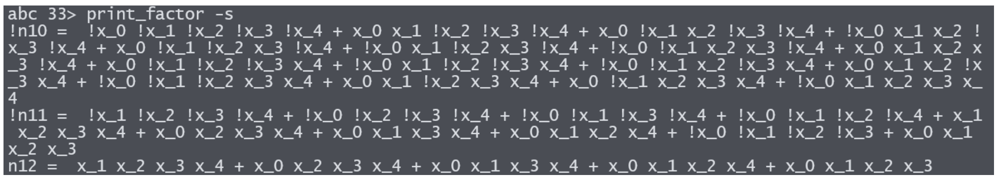
※When running the commands I found that if the *-g* flag is used in *show_bdd*, the node function representation is turned into AIG instead of BDD, which I suspect is not the intended behavior. To address this I simply type *bdd* after using *show_bdd -g*

We can observe that the equivalent multi-level logic network gets transformed into a single-level one by *collapse* but not *bdd*. This is similar to the above case with *aig* and *strash*; *bdd* does not transform the network structure while *collapse* does.

3(b)

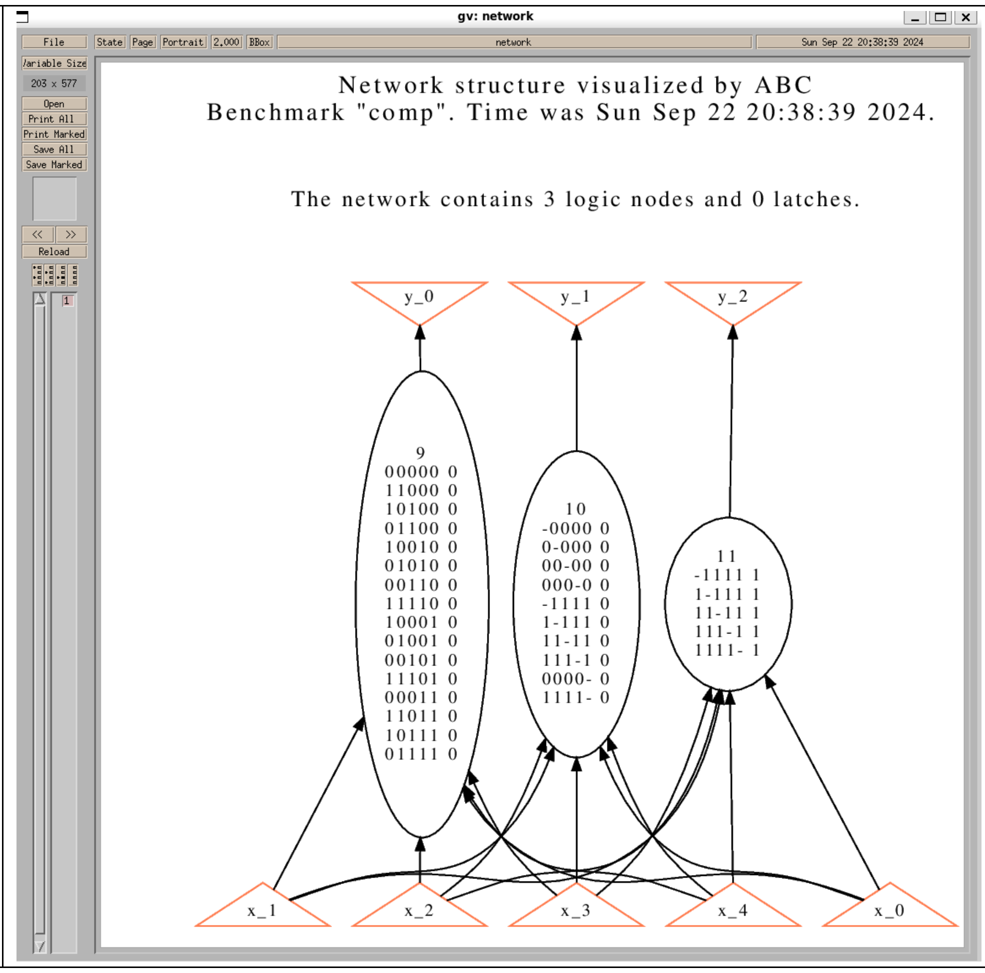
```
read lsv/pa1/comp.blif
strash
                # sequence starts
renode -s       # transform the network back to a logic network
                # -s minimizes the number of SOP cubes
print_factor -s # print the expression of each node
                # it defaults to printing factored forms, -s enables printing
                # SOP
```

Result:



```
abc 33> print_factor -s
!n10 =  !x_0 !x_1 !x_2 !x_3 !x_4 + x_0 x_1 !x_2 !x_3 !x_4 + x_0 !x_1 x_2 !x_3 !x_4 + !x_0 x_1 x_2 !
x_3 !x_4 + x_0 !x_1 !x_2 x_3 !x_4 + !x_0 x_1 !x_2 x_3 !x_4 + !x_0 !x_1 x_2 x_3 !x_4 + x_0 x_1 x_2 x
_3 !x_4 + x_0 !x_1 !x_2 !x_3 x_4 + !x_0 x_1 !x_2 !x_3 x_4 + !x_0 !x_1 x_2 !x_3 x_4 + x_0 x_1 x_2 !x
_3 x_4 + !x_0 !x_1 !x_2 x_3 x_4 + x_0 x_1 !x_2 x_3 x_4 + x_0 !x_1 x_2 x_3 x_4 + !x_0 x_1 x_2 x_3 x_
4
!n11 =  !x_1 !x_2 !x_3 !x_4 + !x_0 !x_2 !x_3 !x_4 + !x_0 !x_1 !x_3 !x_4 + !x_0 !x_1 !x_2 !x_4 + x_1
 x_2 x_3 x_4 + x_0 x_2 x_3 x_4 + x_0 x_1 x_3 x_4 + x_0 x_1 x_2 x_4 + !x_0 !x_1 !x_2 !x_3 + x_0 x_1
x_2 x_3
n12 =  x_1 x_2 x_3 x_4 + x_0 x_2 x_3 x_4 + x_0 x_1 x_3 x_4 + x_0 x_1 x_2 x_4 + x_0 x_1 x_2 x_3
```

show gives



Network structure visualized by ABC
Benchmark "comp". Time was Sun Sep 22 20:38:39 2024.

The network contains 3 logic nodes and 0 latches.

```
9
00000 0
11000 0
10100 0
01100 0
10010 0
01010 0
00110 0
11110 0
10001 0
01001 0
00101 0
11101 0
00011 0
11011 0
10111 0
01111 0
```

```
10
-0000 0
0-000 0
00-00 0
000-0 0
-1111 0
1-111 0
11-11 0
111-1 0
0000- 0
1111- 0
```

```
11
-1111 1
1-111 1
11-11 1
111-1 1
1111- 1
```