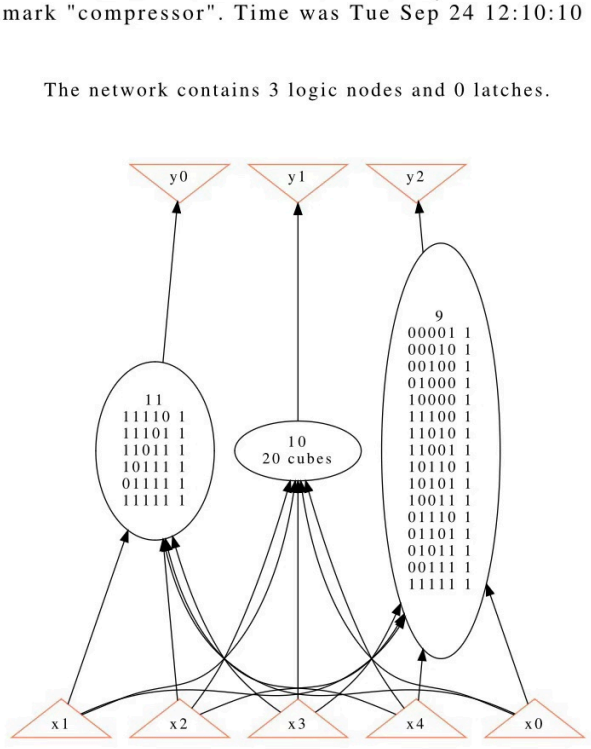
(a) Create a BLIF file named "`comp.blif`" to represent a 5-to-3 compressor with 5 inputs x_0, x_1, x_2, x_3, and x_4. Its output $Y = (y_2, y_1, y_0)$ is a 3-bit unsigned integer that represents the number of 1's in the inputs. For example, when inputs are 10011, output Y = $(011)_2$, i.e., binary number of 3, because there are three 1's in 10011. You can find the BLIF manual in `http://www.eecs.berkeley.edu/~alanmi/publications/other/blif.pdf`.

(b) Perform the following steps to practice using ABC with your "`comp.blif`". Screenshot the results after running the commands and put them in your report.

1. read the BLIF file into ABC (command "`read`")
2. check statistics (command "`print_stats`")
3. visualize the network structure (command "`show`")
4. convert to AIG (command "`strash`")
5. visualize the AIG (command "`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 01> read lsv/pa1/comp.blif
Warning: The design has 2 root-level modules. The first one (compressor) will be used.
abc 02> print_stats
compressor                 : i/o =    5/    3  lat =    0  nd =     3  edge =     15  cube =    42  lev = 1
abc 02> show
abc 02> gv: Unable to open the display.
abc 02> strash
abc 03> print_stats
compressor                 : i/o =    5/    3  lat =    0  and =     59  lev =  8
abc 03> show
abc 03> gv: Unable to open the display.
abc 03> collapse
abc 04> print_stats
compressor                 : i/o =    5/    3  lat =    0  nd =     3  edge =     15  bdd  =    21  lev = 1
abc 04> show_bdd
abc 04> gv: Unable to open the display.
abc 04> show_bdd
abc 04> gv: Unable to open the display.
abc 04> show_bdd -g
abc 04> gv: Unable to open the display.
```

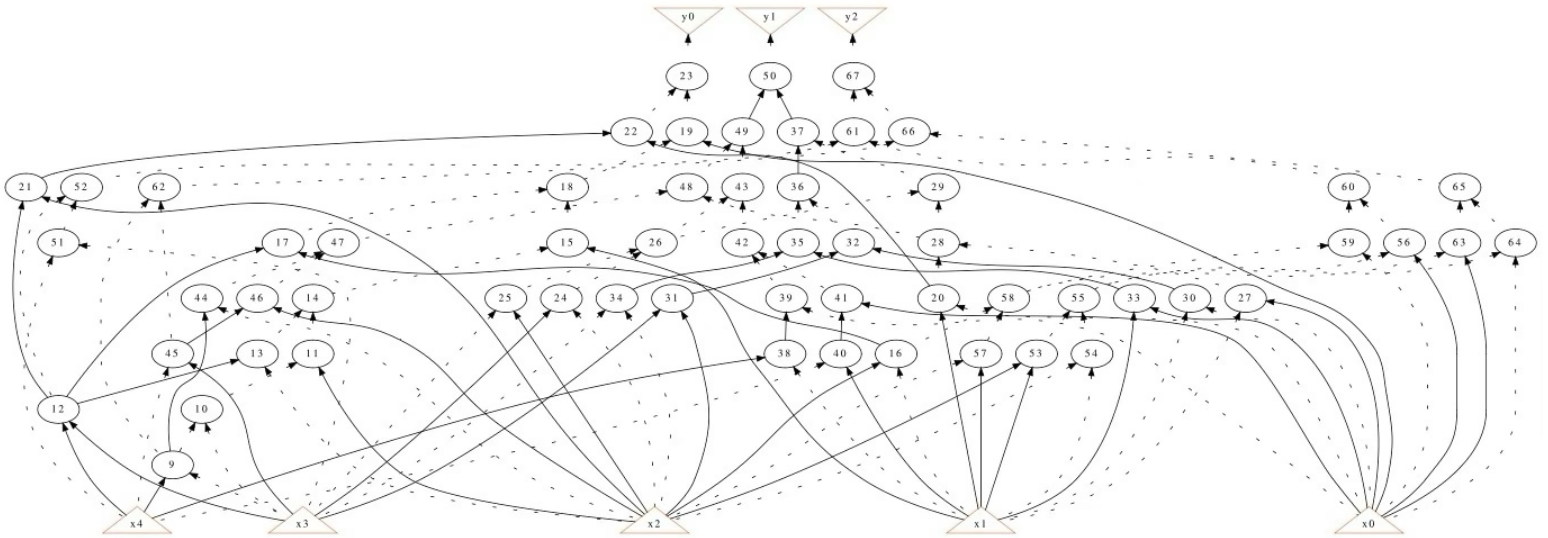
3.



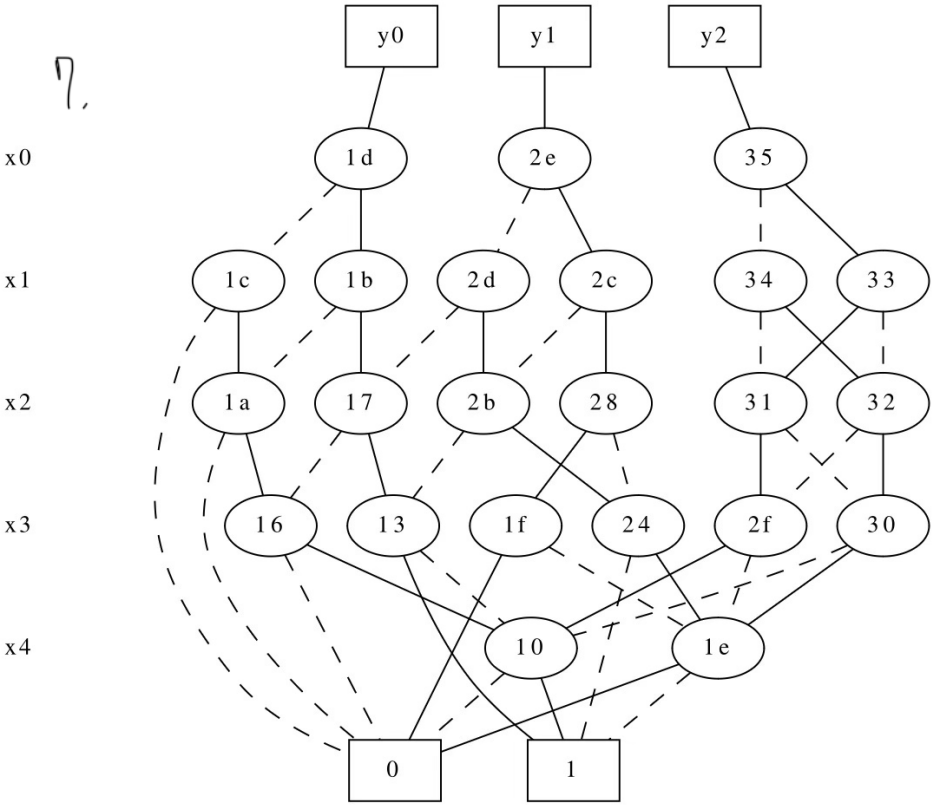
5.

Network structure visualized by ABC
Benchmark "compressor". Time was Tue Sep 24 12:12:45 2024.

The network contains 59 logic nodes and 0 latches.



7.



3 [ABC Boolean Function Representations] (10%)

In ABC, there are different ways to represent Boolean functions.

(a) Compare the following differences with your "`comp.blif`". Screenshot the results and briefly describe your findings in your report.
   1. logic network in AIG (by command "`aig`") vs. structurally hashed AIG (by command "`strash`")
   2. logic network in BDD (by command "`bdd`") vs. collapsed BDD (by command "`collapse`")

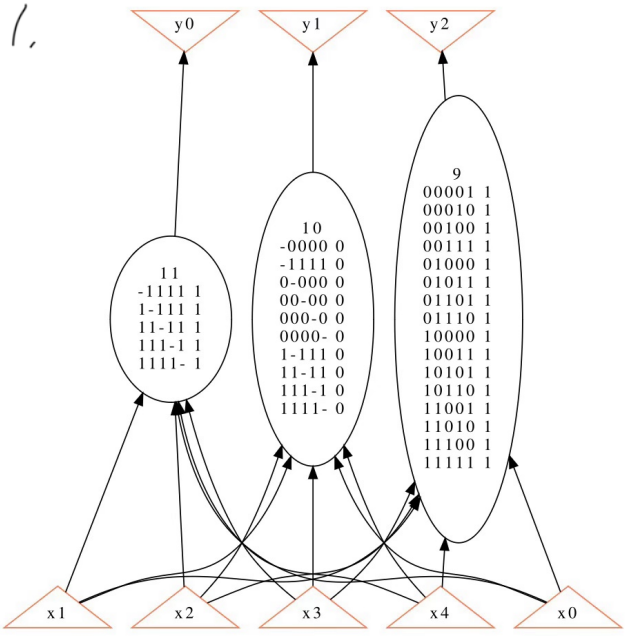
(b) Given a structurally hashed AIG, find a sequence of ABC commands to convert it to a logic network with node function expressed in sum-of-products (SOP). Use your "`comp.blif`"to test your command sequence (by first running "`strash`" to convert it to AIG). Screenshot the results, and put them in your report.

(a)

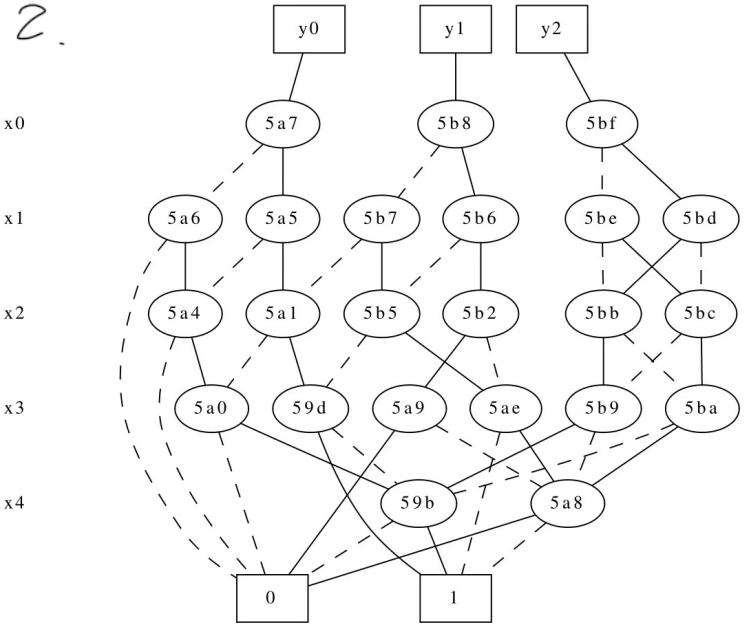
```
abc 01> read lsv/pa1/comp.blif
Warning: The design has 2 root-level modules. The first one (compressor) will be used.
abc 02> aig
abc 02> show
abc 02> gv: Unable to open the display.

abc 02> show
abc 02> gv: Unable to open the display.
abc 02> bdd
abc 02> show_bdd
abc 02> gv: Unable to open the display.
abc 02> show_bdd -g
abc 02> gv: Unable to open the display.
```

1.



2.



The command "aig" didn't do anything to the original network, whereas "strash" do convert the original network into AIG

These two commands have no difference except for the name of the node

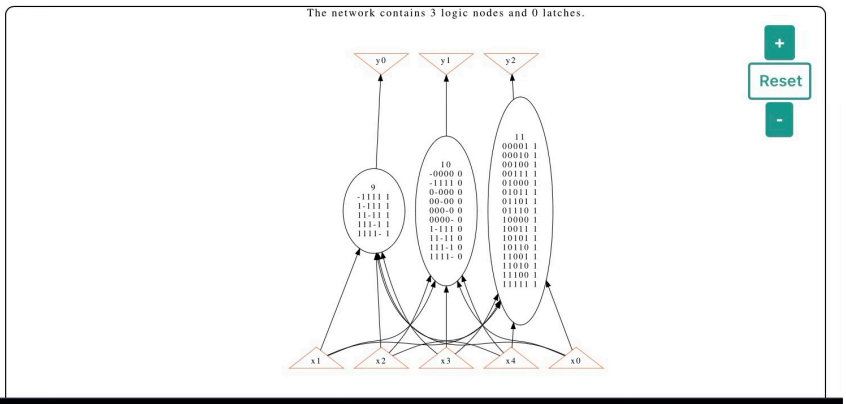
(b)

compressor.ps

PostScript Preview - compressor.ps

compressor.ps

Hide Controls



問題 輸出 偵錯主控台 終端機 連接埠 註解

```
abc 01> read lsv/pa1/comp.blif
Warning: The design has 2 root-level modules. The first one (compressor) will be used.
abc 02> strash
abc 03> collapse
abc 04> sop
abc 04> show
abc 04> gv: Unable to open the display.
abc 04>
```