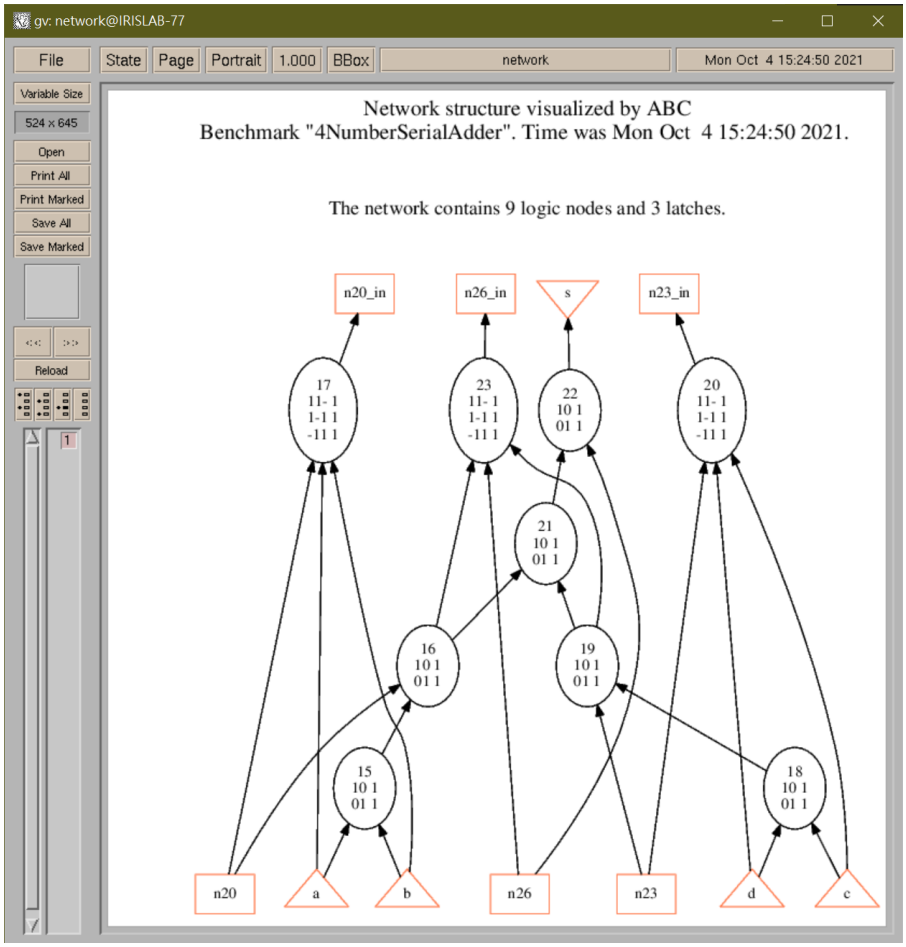
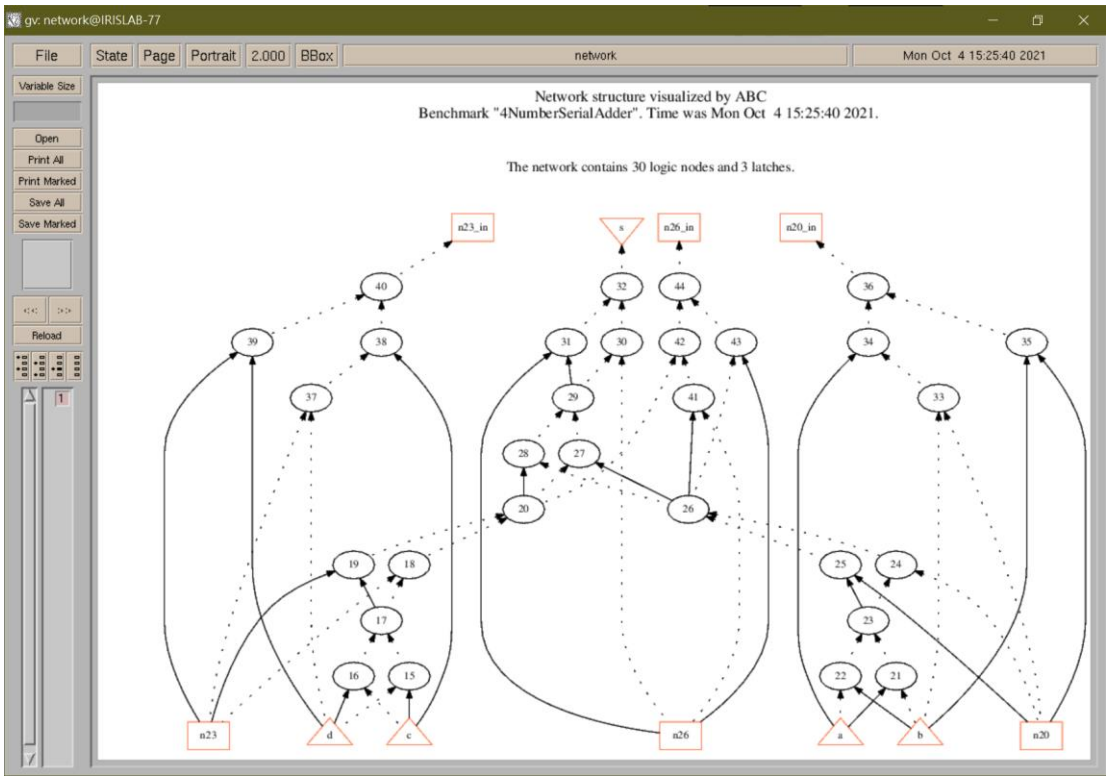
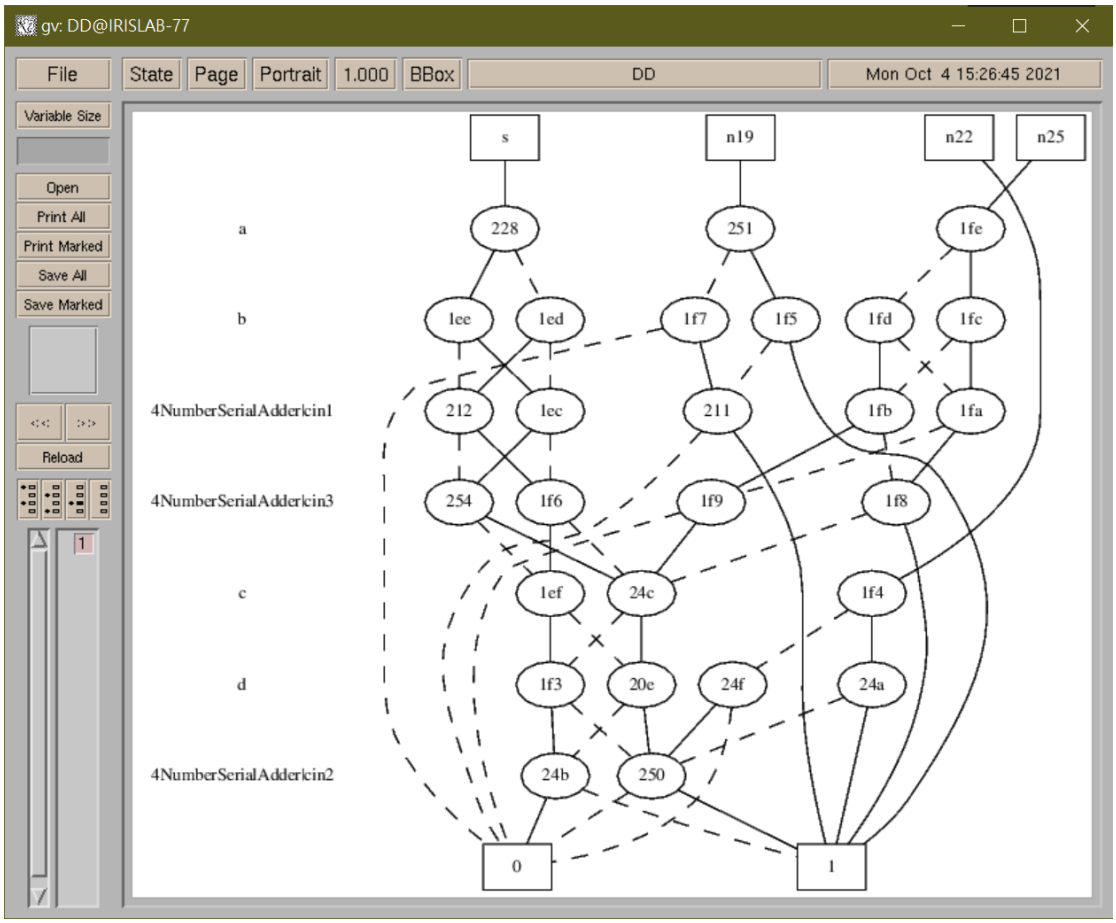
PA1-part1-b-3 (“show” network structure)



PA1-part1-b-5 (“show” aig)



PA1-part1-b-7 (“show_bdd -g” bdd)



PA1-part2-a

logic network in AIG (by command aig) is the origin node structure with all of node's internal logic's data structure is transformed to aig. When we command "show", the structure of the nodes will be the same as the original one. But inside the node, the data structure has been changed.

structurally hashed AIG (by command strash) is ripping up all of the nodes and transform the logic's data structure to aig. When we command "show", the result should have more nodes and more levels, and each node should be an aig.

logic network in BDD (by command bdd) is the origin node structure with all of node's internal logic's data structure is transformed to bdd. When we command "show", the structure of the nodes will be the same as the original one. But inside the node, the data structure has been changed.

collapsed BDD (by command collapse) is collapsing all of the PI/PPIs to one bdd node representing one PO/PPO. When we command "show", the result should have only one level with more complex logic each node, one PO/PPO corresponding to one node, and each node should be a bdd as its data structure.

print_stats of each command

origin

```
4NumberSerialAdder          : i/o =    4/    1  lat =    3  nd =     9  edge =     21  cube =    21  lev = 4
```

aig

```
4NumberSerialAdder          : i/o =    4/    1  lat =    3  nd =     9  edge =     21  aig  =    30  lev = 4
```

strash

```
4NumberSerialAdder              : i/o =    4/     1  lat =     3  and =      30  lev =  8
```

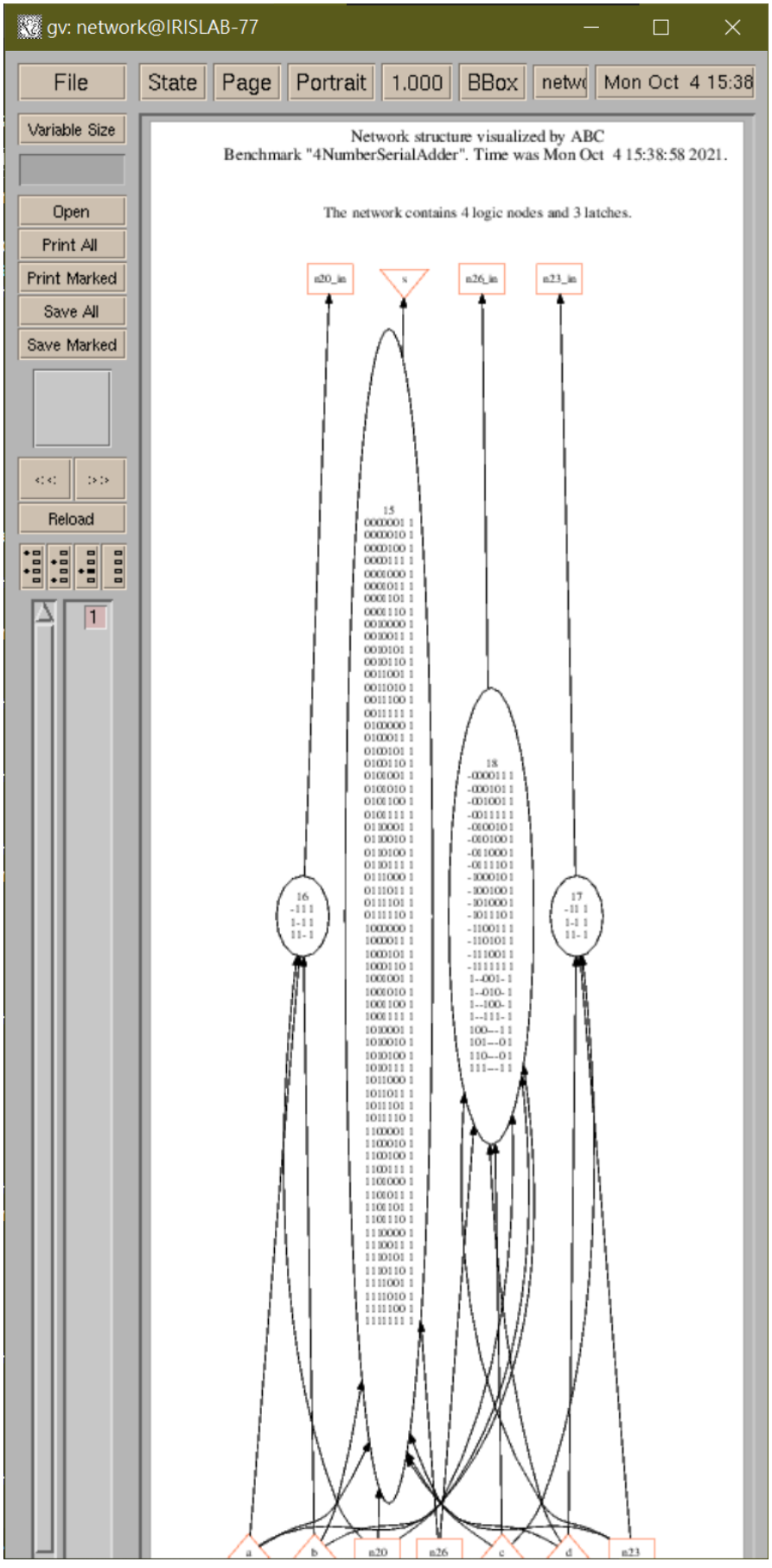
bdd

```
4NumberSerialAdder          : i/o =    4/    1  lat =    3  nd =     9  edge =     21  bdd  =    24  lev = 4
```

collapse

```
4NumberSerialAdder          : i/o =    4/    1  lat =    3  nd =     4  edge =     20  bdd  =    25  lev = 1
```

“show” after collapse



PA1-part2-b
“renode”+“sop”
(or “logic”)

usage: renode [-KCFA <num>] [-sbciav]
transforms the AIG into a logic network with larger nodes
while minimizing the number of FF literals of the node SOPs

usage: sop [-C num] [-sdnh]
converts node functions to SOP

usage: logic [-h]
transforms an AIG into a logic network with SOPs