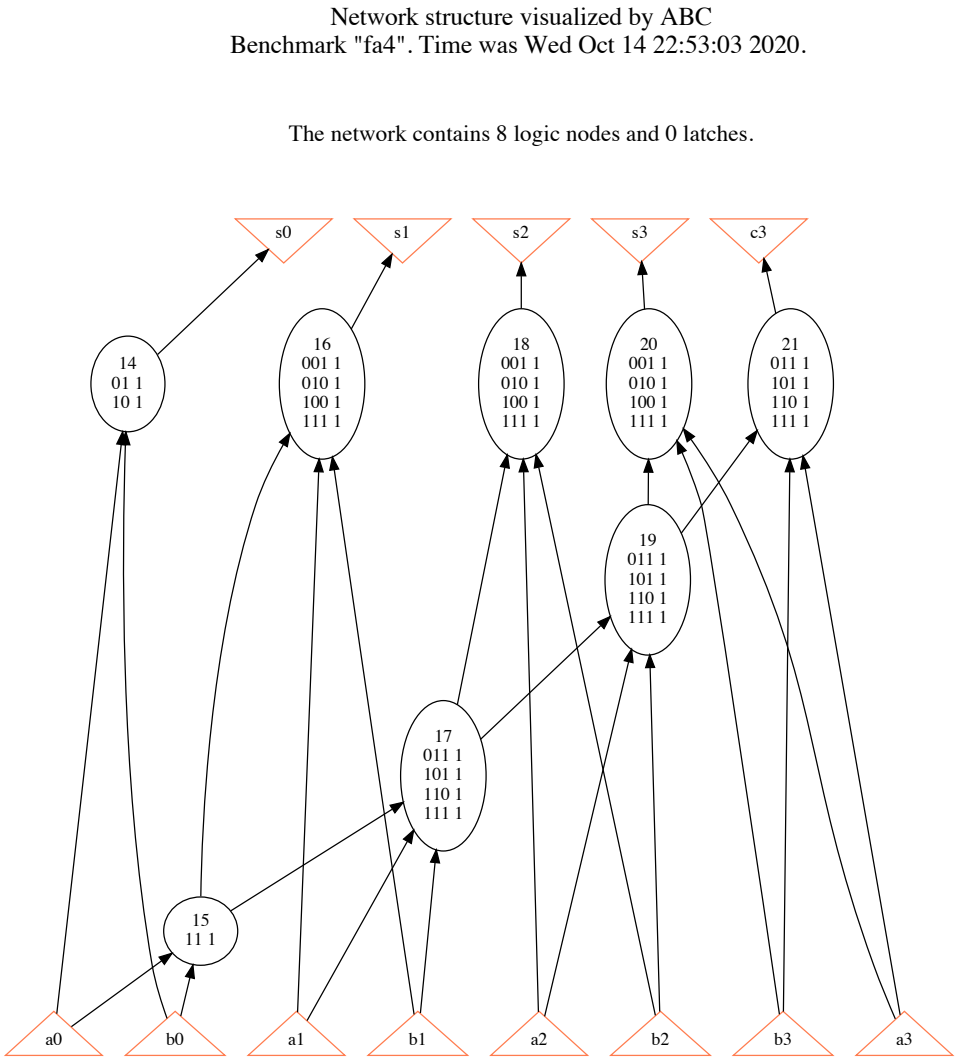
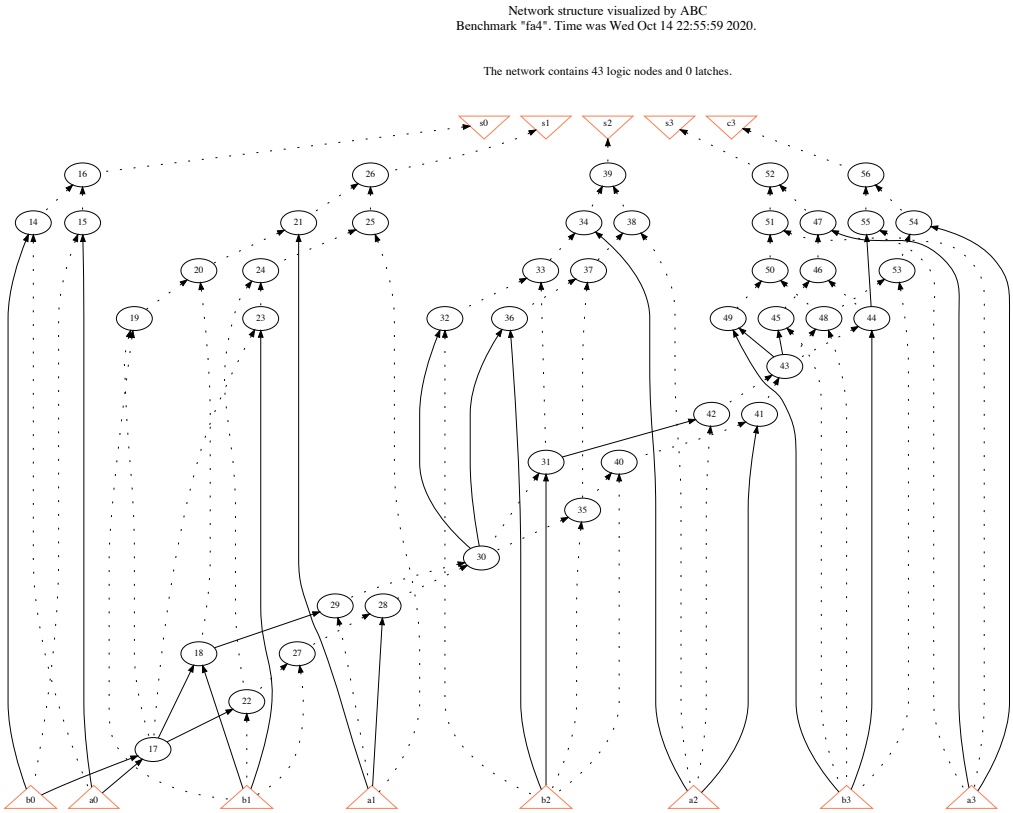
I let the four-bit adder input be $a_0a_1a_2a_3$, and $b_0b_1b_2b_3$. I let the output be four-bit sum $s_0s_1s_2s_3$ and one-bit carry c_3.

Part 1

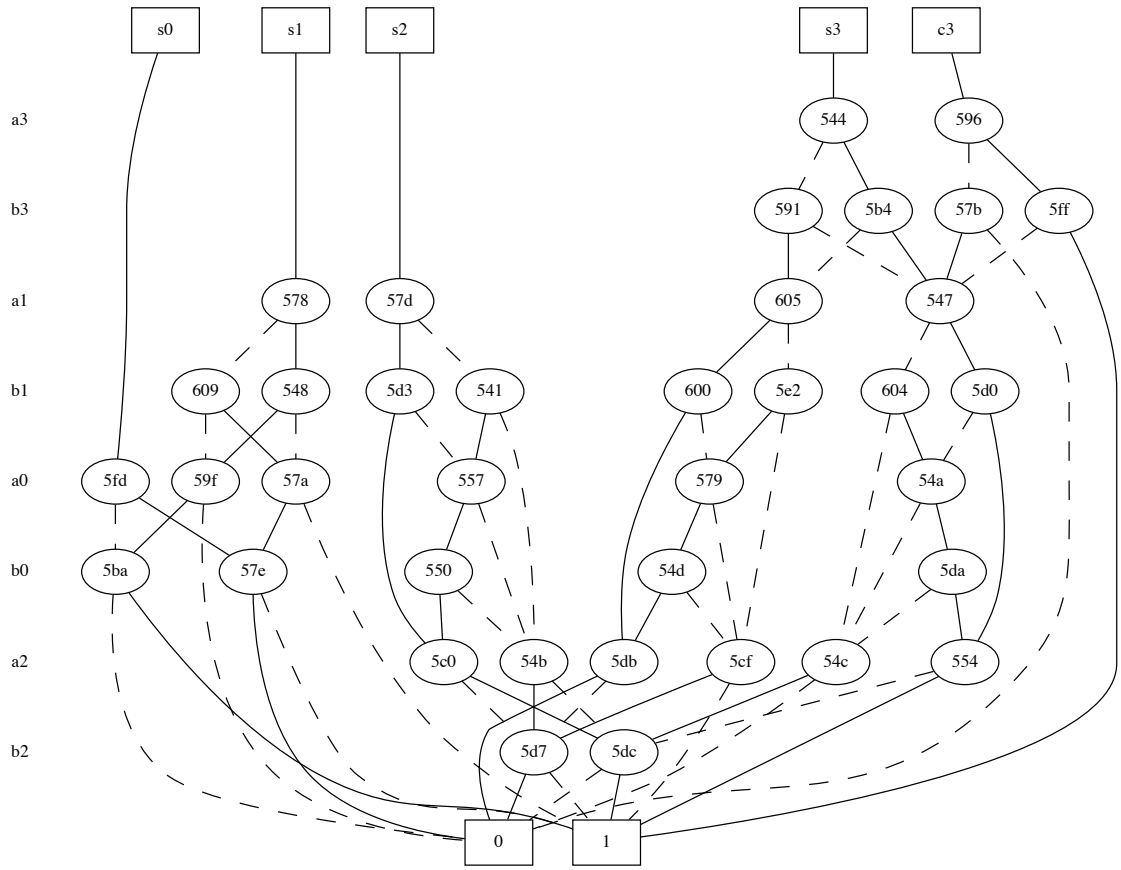
1. logic network



2. Strash


Network structure visualized by ABC
Benchmark "fa4". Time was Wed Oct 14 22:55:59 2020.
The network contains 43 logic nodes and 0 latches.
s0
s1
s2
s3
c3
16
26
39
52
56
14
15
21
25
34
38
51
47
55
54
20
24
33
37
50
46
53
19
23
32
36
49
45
48
44
43
42
41
31
40
35
30
29
28
18
27
22
17
b0
a0
b1
a1
b2
a2
b3
a3

3. Collapse


s0
s1
s2
s3
c3
a3
b3
a1
b1
a0
b0
a2
b2
544
596
591
5b4
57b
5ff
578
57d
605
547
609
548
5d3
541
600
5e2
604
5d0
5fd
59f
57a
557
579
54a
5ba
57e
550
54d
5da
5c0
54b
5db
5cf
54c
554
5d7
5dc
0
1

Part 2

a) Compare the following differences with the four-bit adder example.

1. logic network in AIG (by command aig) vs. structurally hashed AIG (by command strash)

   The information is printed by typing -h after the command in abc

   - "aig -h": converts node functions to AIG. (The entire circuit is still represented by logic network, but the node functions are represented by AIG)
   - "strash -h": transforms combinational logic into an AIG. (The entire circuit is represented by AIG)

2. logic network in BDD (by command bdd) vs. collapsed BDD (by command collapse)

   - "collapse -h": collapses the network by constructing global BDDs. (The entire circuit is represented by BDD)
   - "bdd -h": converts node functions to BDD. (The entire circuit is still represented by logic network, but the node functions are represented by BDD)

b) Given a structurally hashed AIG, find a sequence of ABC command(s) to covert it to a logic network with node function expressed in sum-of-products (SOP).

   - The command "logic -h": Given a structurally hashed AIG, find a sequence of ABC command(s) to covert it to a logic network with node function expressed in sum-of-products (SOP).