Designing Oracles for Grover Algorithm

Homework

1. You have time until December 3 to return me this homework.

2. Please use PPT, Word or some word processor. You may send also PDF. The simulation should be done in Matlab. Matlab code and all results should be submitted with comments and explanations.

3. The homework is to simulate Grover algorithm on some small oracle of your design.

4. The absolute minimum is to solve the map coloring problem (planar graph coloring) which I discussed in the class. You have to show all stages of your design and explanations of design procedures. For doing this part of homework you obtain 10 points.

5. If you will add a part of oracle to calculate that the cost of coloring is below certain value T, you will obtain another 10 points. See some hints in next slides.

6. If you design an oracle for some other NP-hard or NP-complete problem, different than the graph coloring problem, and simulate it successfully, you will obtain a total of 20 points. This cannot be a satisfiability problem and the problem should have some practical meaning and not be some arbitrary Boolean function as an oracle.

7. Quality of presentation of results, explanation, figures and use of English will be also taken into account while grading.

Examples of Problems for Oracles

- Satisfiability oracles: These oracles are based on creating the single-output satisfiability formula. The formula can use various gate types and structures, depending on the problem.

- Constraint satisfaction oracles: These type of oracles are for constraint satisfaction problems such as graph coloring, image matching or cryptographic puzzles. These oracles use both logical, arithmetical and relational blocks and have often the decision oracle and the optimization oracle as their components. The decision oracle is the global AND of several partial decision sub-oracles.

- Path problems: These are problems to find certain path in a graph, for introduce Euler path or Hamiltonian path. Many games and puzzles such as "Man, wolf, Goat and Cabbage" belong to this category. The oracles includes decision sub-oracles for each move(edge) in the graph of the problem(game)

- Problems related to spectral transforms:

- The mapping problems, including their special class, the subset selection problems.

The Satifiability oracles includes the following:

- POS satisfiability.
- Solving unite covering problem by Petrich Function.
- Solving binate covering problem
- Solving various multi-level SAT formulas, especially the generalized SAT of the form
- Solving even-odd covering problem for ESOP, PPRM, FPRM and similar minimization problems
- Solving AND-OR DAG from robotics and Artificial intelligence.

The constraint satisfiability oracles include:

- Proper graph coloring
- Compatible graph coloring
- Graph coloring problems with non-standard cost functions
- Waltz algorithm for image matching
- Cryptoarithmetic puzzles such as SEND + MORE = MONEY

The Mapping oracles include:

- Maximum cliques (used in Maghoute algorithm for graph coloring)
- Maximum independent set
- Finding prime implicants of the Boolean Function.

<u>Path oracles</u> include:

- Euler path
- Hamilton path
- Shortest path
- Longest path
- Traveling salesman
- Missionaries and cannibals logic puzzle
- Man, Wolf, Goat and Cabbage logic puzzle

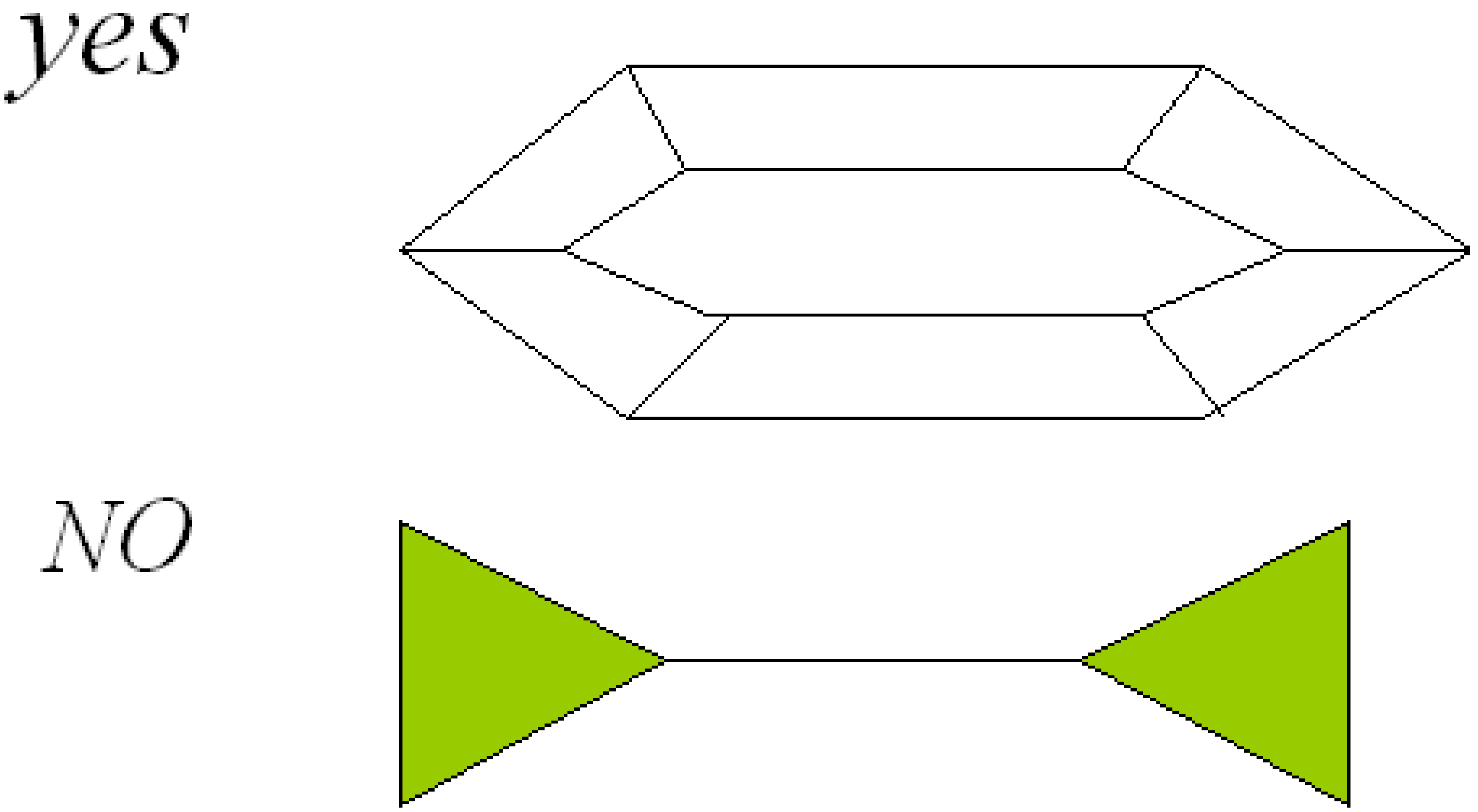
<u>Exhaustive solving</u> of equations includes:

- $a^n + b^n = c^n$

Finding a Hamiltonian Path in a Graph

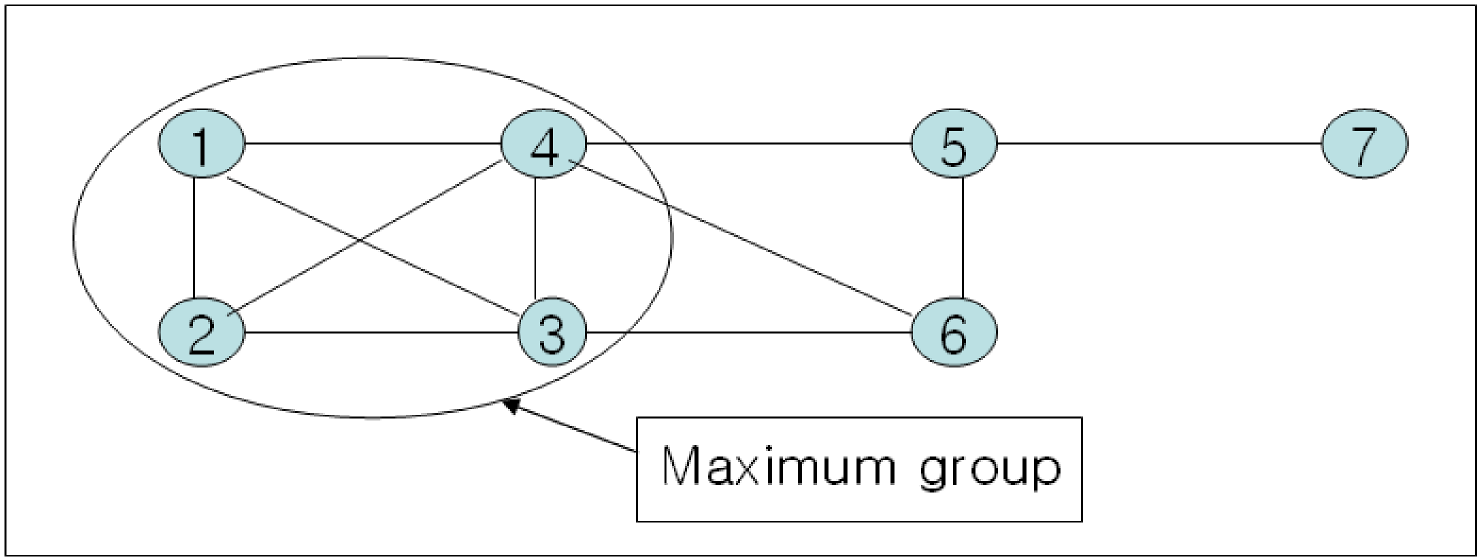
- **Problem.**
- Given is a non-oriented graph. Find a path of edges of this graph that goes through all nodes, passing each of them only once, and finishes in the initial node.

- Find a Hamiltonian path or prove that such path does not exist for a given graph.



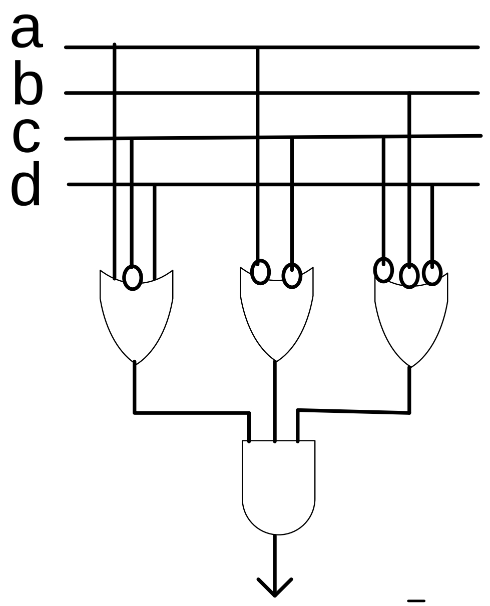
- *(a) A graph with Hamiltonian Path, (b) a graph with no Hamiltonian Path*

Finding Maximum Cliques in a graph



- ***Maximum Clique in a graph.***
- ***There are other maximum cliques but this is the only one maximum clique with four nodes.***
- ***{3, 4, 6} is a maximum clique with 3 nodes.***

Solving the Satisfiability Class of Problems

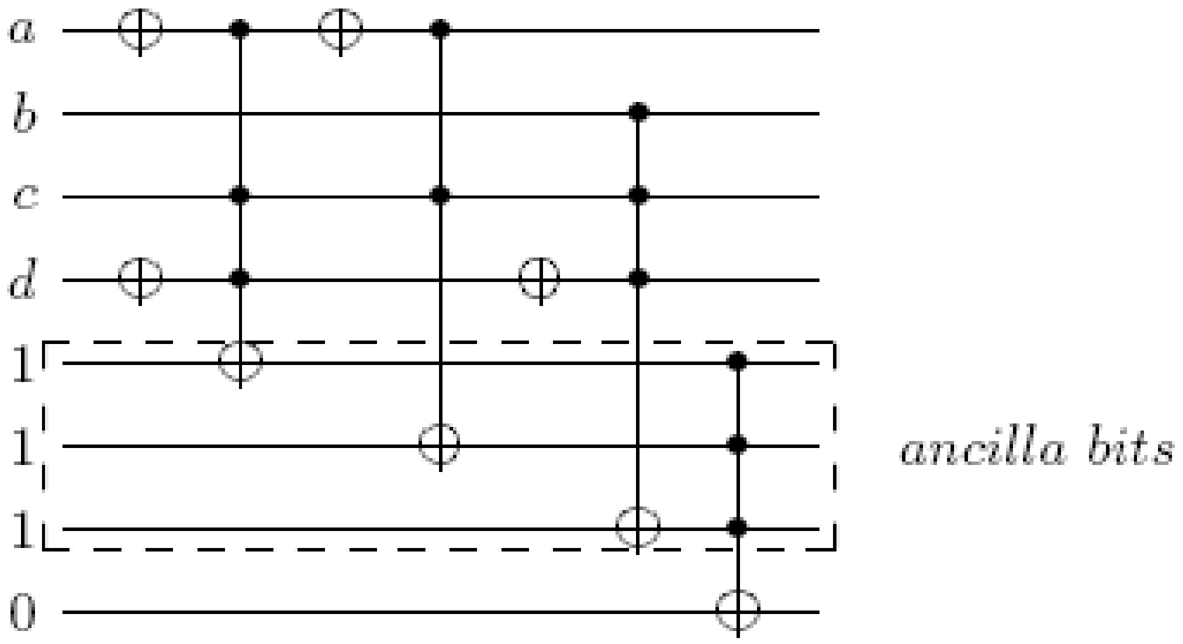


$$f_1 = (a+\bar{c}+d)(\bar{a}+\bar{c})(\bar{c}+\bar{b}+\bar{d})$$

$$\overline{\overline{(a+\bar{c}+d)}} \bullet \overline{\overline{(\bar{a}+\bar{c})}} \bullet \overline{\overline{(\bar{c}+\bar{b}+\bar{d})}}$$

$$= \overline{\bar{a}c\bar{d}} \bullet \overline{ac} \bullet \overline{cbd}$$

- *Classical oracle for POS Satisfiability .*



- *Realization of oracle for POS SAT .*

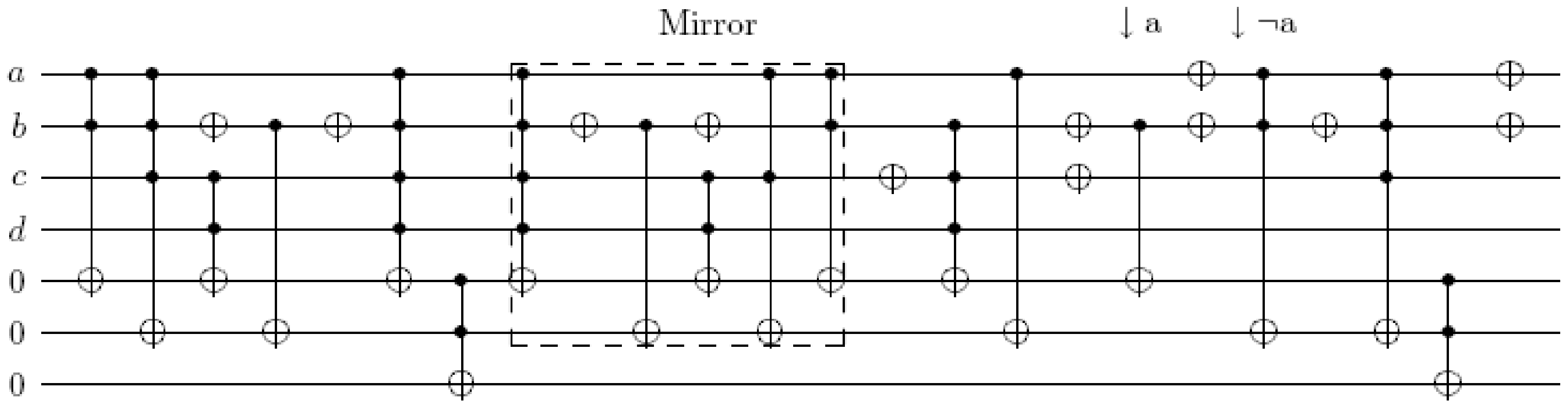
$$f = (a + \bar{c} + d) \bullet (\bar{a} + \bar{c}) \bullet (\bar{c} + \bar{b} + \bar{d})$$

For example, given is a SAT formula:

$$f_2 = [(ab + cd) \bullet (ac + \bar{b})] \oplus [(ab\bar{c}d) \bullet (a + b + c)]$$

The formula is transformed to the following form

$$f_2 = [(ab \oplus cd \oplus abcd) \bullet (\bar{b} \oplus bac)] \oplus [(\bar{b} \oplus ab\bar{c}d) \bullet (a \oplus \bar{a}b \oplus \bar{a}\bar{b}c)]$$

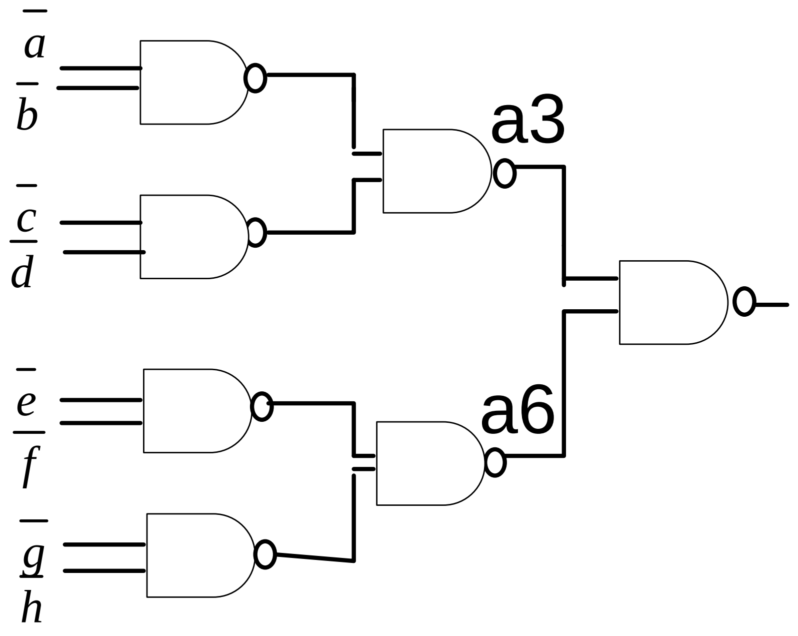


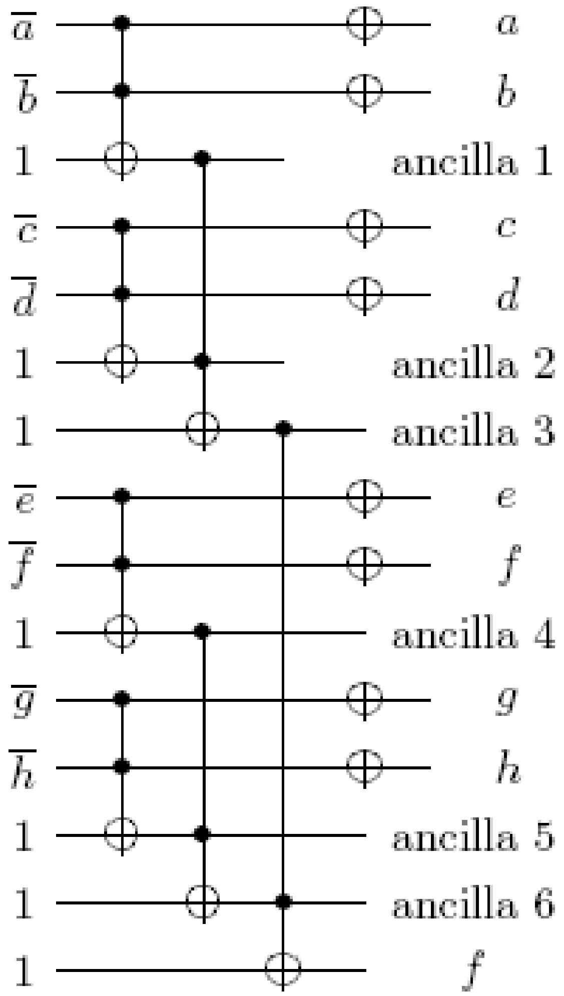
Oracle for function $f_2 = [(ab + cd) \bullet (ac + \bar{b})] \oplus [(ab\bar{c}d) \bullet (a + b + c)]$

using mirror circuits to decrease the number of ancilla bits.

The circuit is not minimized.

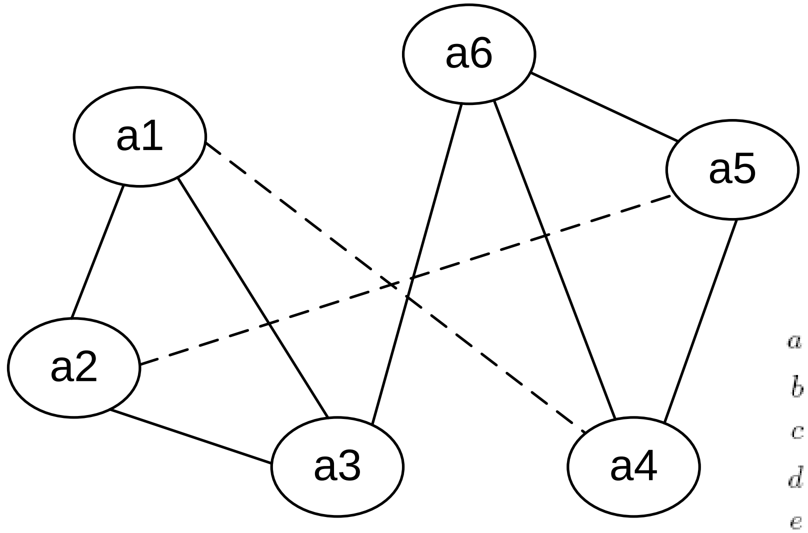
Classical versus Quantum Oracle





B) Non optimized quantum array of the classical oracle from Fig A)

Maximum Cliques and Maximum Independent Sets of graphs



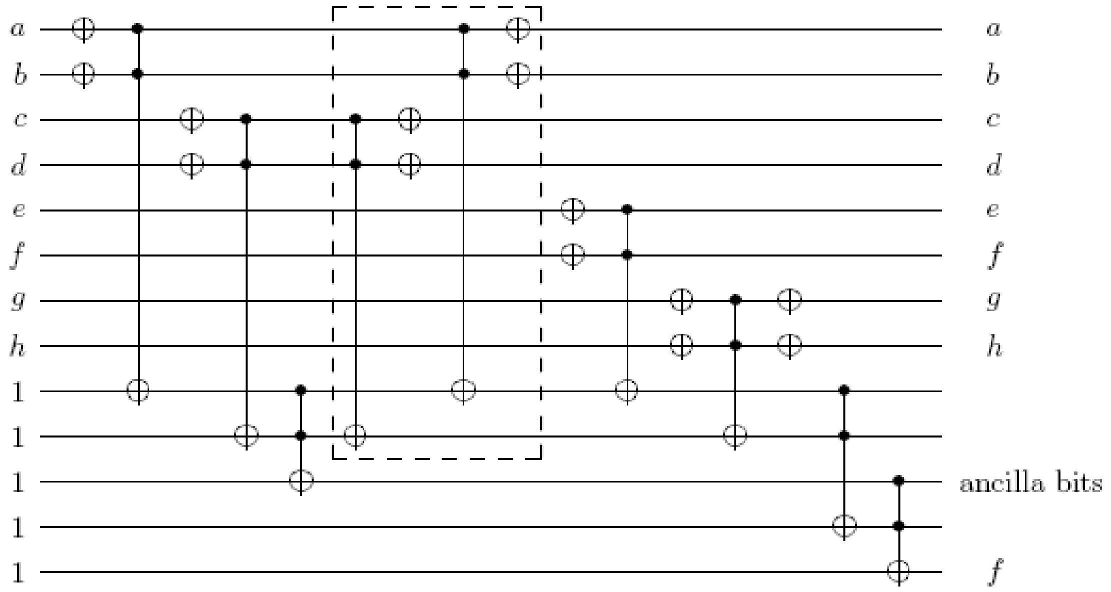


Fig. A presents the incompability graph.
Every two ancilla bits that can not be combined are linked by a solid edge.
The graph shows that there are the following maximum
independent sets: {a_1, a_4}, { a_1, a_5}, { a_2, a_5}, { a_2 , a_4}, among others.
We select pairs { a_1, a_4} and { a_2 , a_5} for folding.
This leads to the quantum array with mirror circuit, presented in Fig. B

CONSTRAINTS SATISFACTION PROBLEMS

$$\begin{array}{r} S\,E\,N\,D \\ \underline{+\;\; M\,O\,R\,E} \\ M\,O\,N\,E\,Y \end{array}$$

- Fig. 9.9.

$$\begin{array}{ll}
D + E = 10\,C_1 + Y & C_1\ \{0, 1\} \\
N + R + C_1 = 10\,C_2 + E & C_2\ \{0, 1\} \\
E + O + C_2 = 10\,C_3 + N & C_3\ \{0, 1\} \\
S + M + C_3 = 10\,C_4 + O & C_4\ \{0, 1\} \\
C_4 = M &
\end{array}$$

Fig. 9.10a. Equations compiled from the problem formulation from Figure 9.9.

S { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
E { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
N { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
D { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
M { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
O { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
R { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
Y { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

Fig. 9.10b. Constraints for nodes in the graph.

$S \neq E$, $S \neq N$, $S \neq D$, $S \neq M$ etc.

Fig. 9.10c. Inequalities for unique encoding of nodes of the graph.

$C_4 = M$ $C_4 \{ 0, 1 \}$

$M = 1$

$S + M + C_3 = 10\, C_4 + O$

$S + M + C_3 = 10\, M + O$

$S + C_3 = 9\, M + O$

$S + C_3 = 9 + O$ (Simplified equation)

Fig.9.11. Equations compiled from the SEND+MORE=MONEY equation

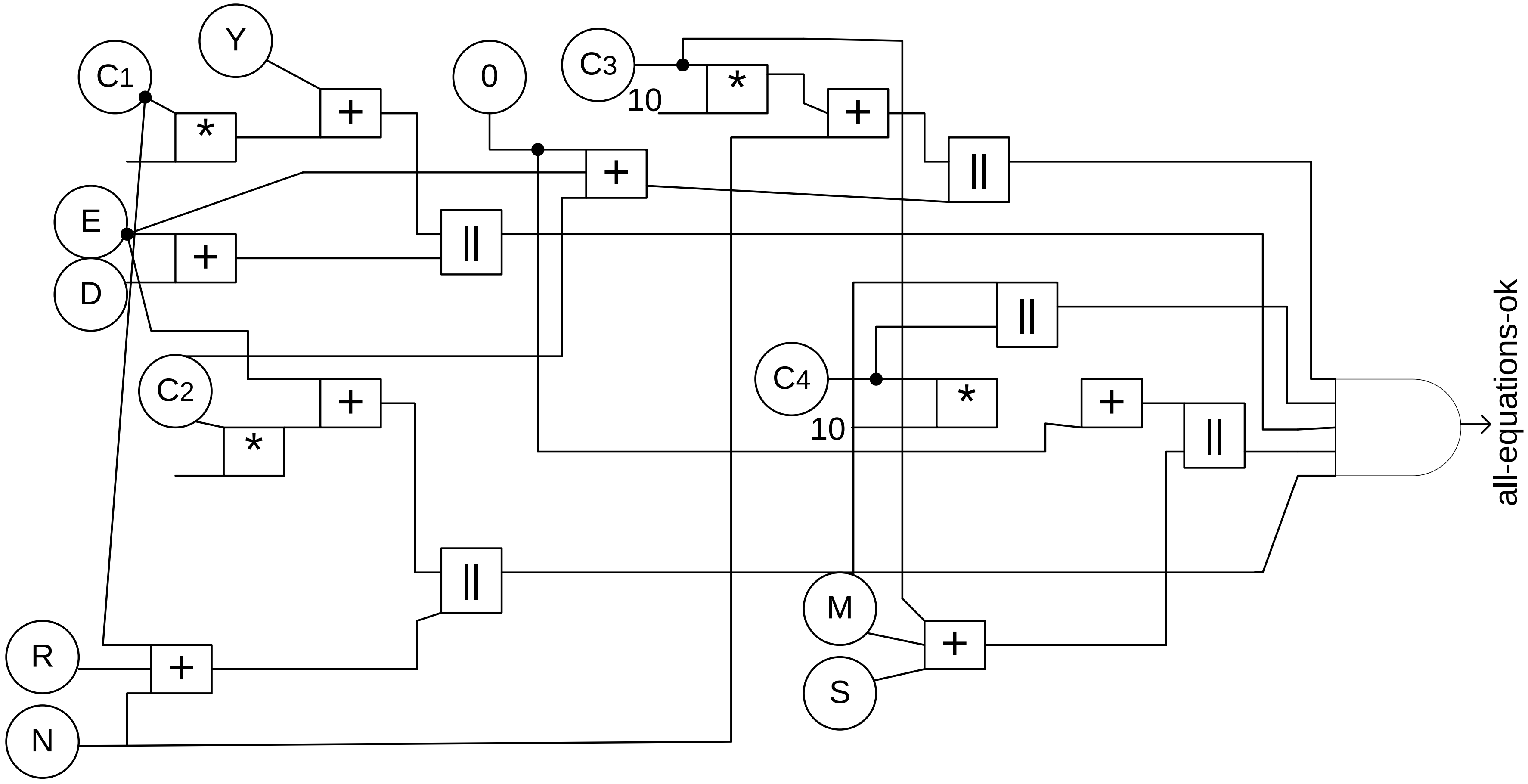


Fig. 9.12. Graph of constraints for the SEND+MORE=MONEY problem.

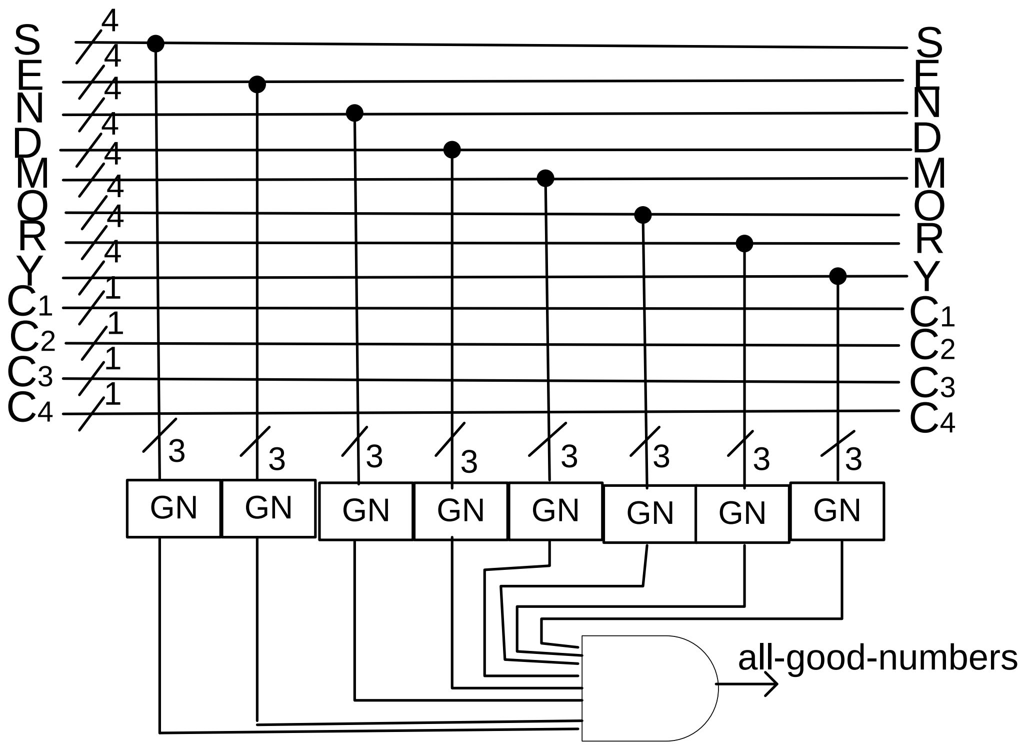
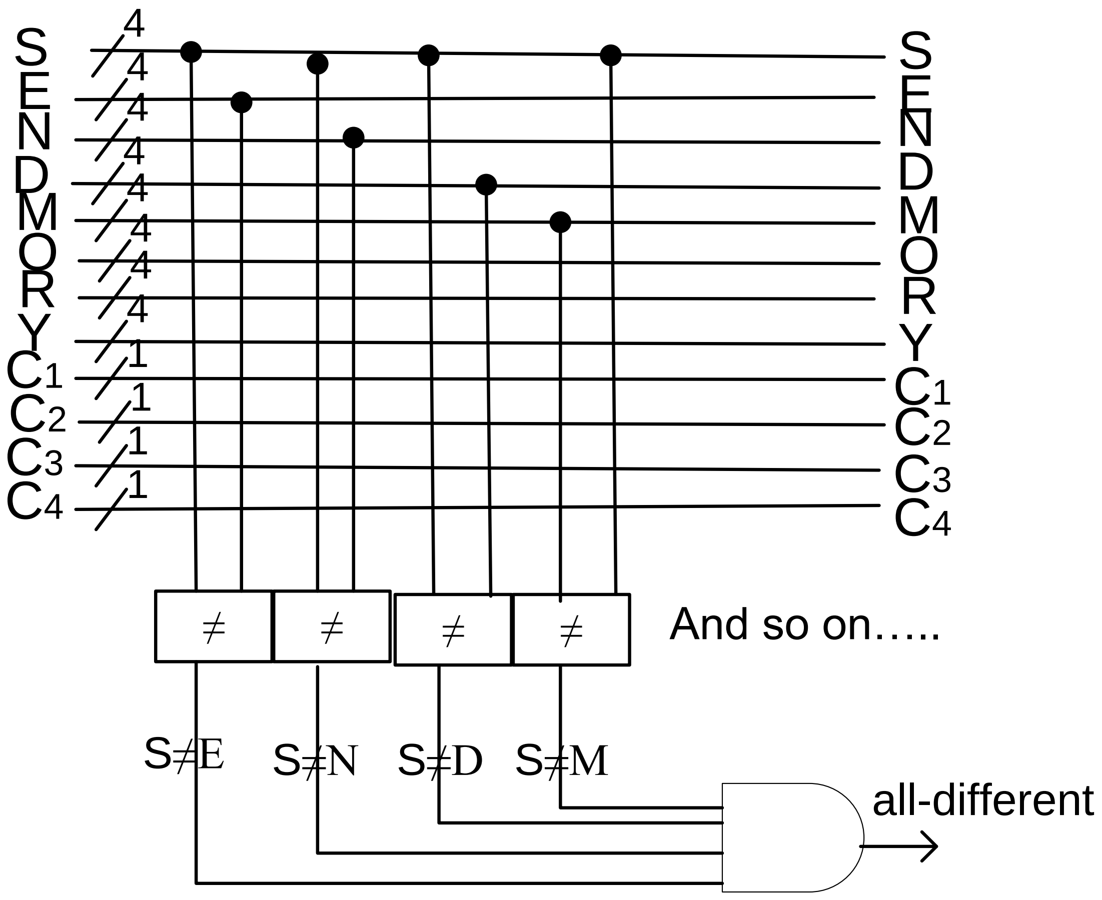


Fig. 9.14. The remaining part of the oracle for the SEND+MORE=MONEY problem.



- *Fig.9.15. The part of an oracle for the SEND+MORE=MONEY problem that checks uniqueness of mapping*

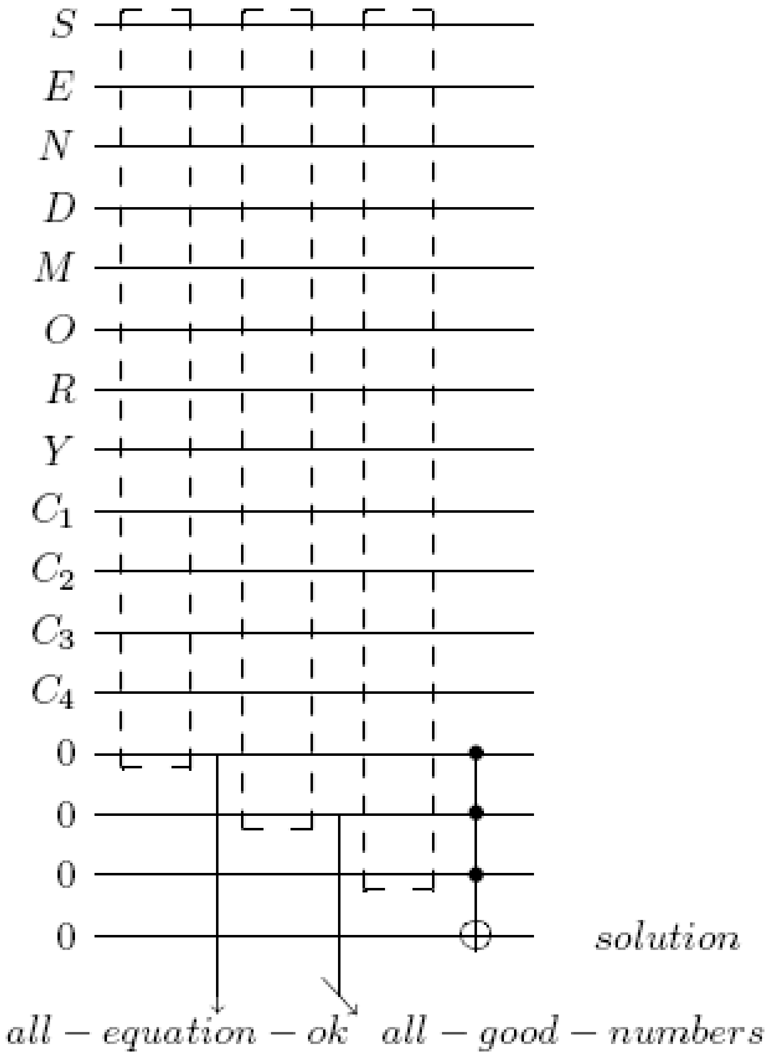
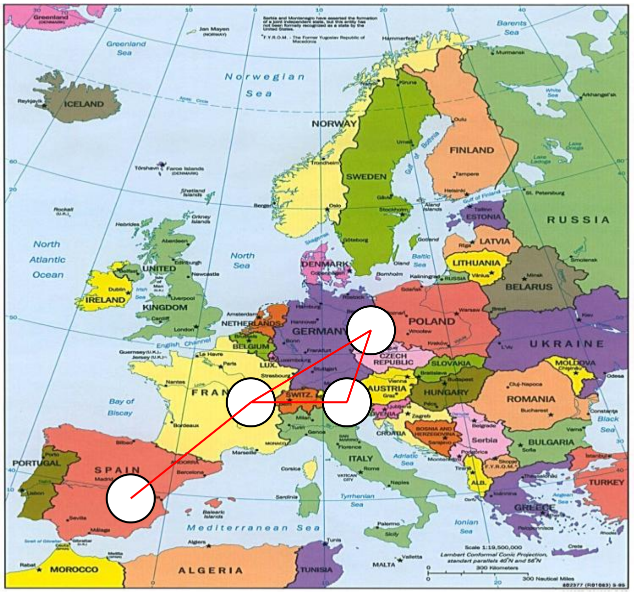


Fig. 9.16. The part of oracle for the SEND+MORE=MONEY problem that….



ORACLES FOR CONSTRAINT SATISFACTION PROBLEMS

Map of Europe

The Graph Coloring Problem.

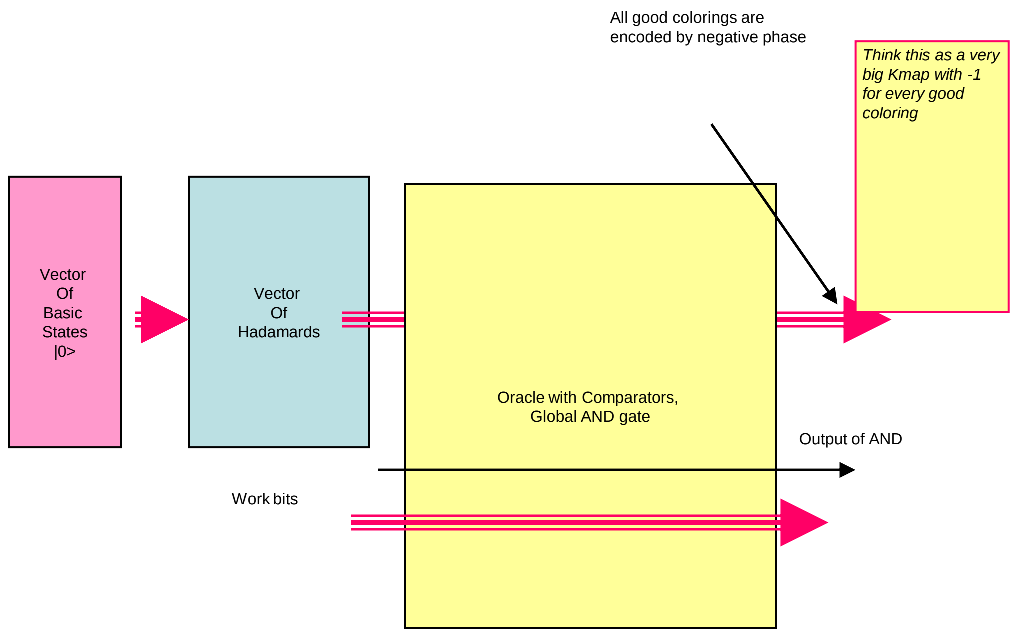

All good colorings are encoded by negative phase
Think this as a very big Kmap with -1 for every good coloring
Vector Of Basic States |0>
Vector Of Hadamards
Oracle with Comparators, Global AND gate
Output of AND
Work bits

Figure 5: Block Diagram

Figure 6a: A simple graph coloring problem

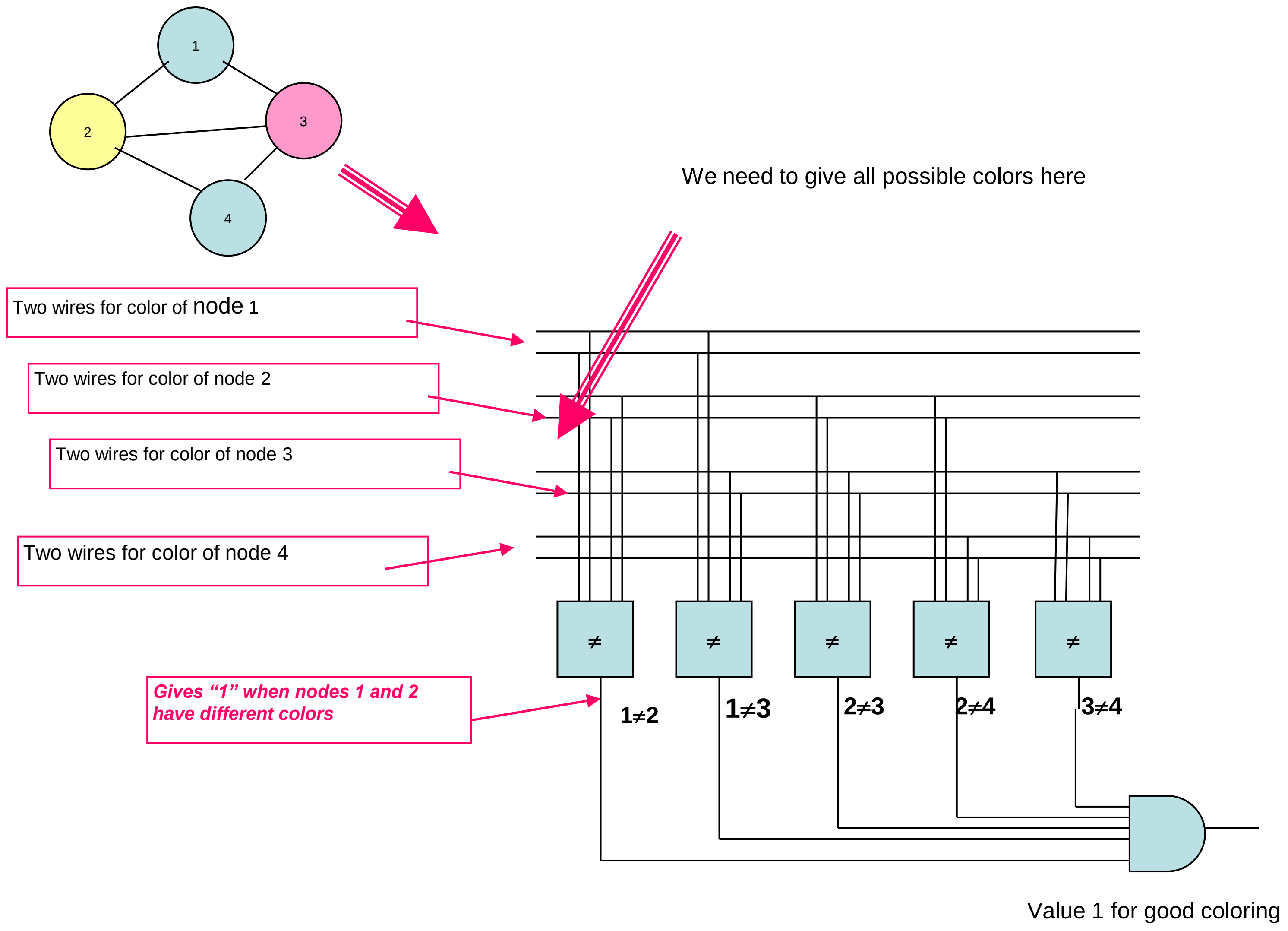
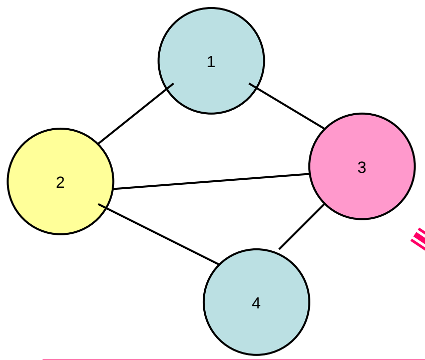
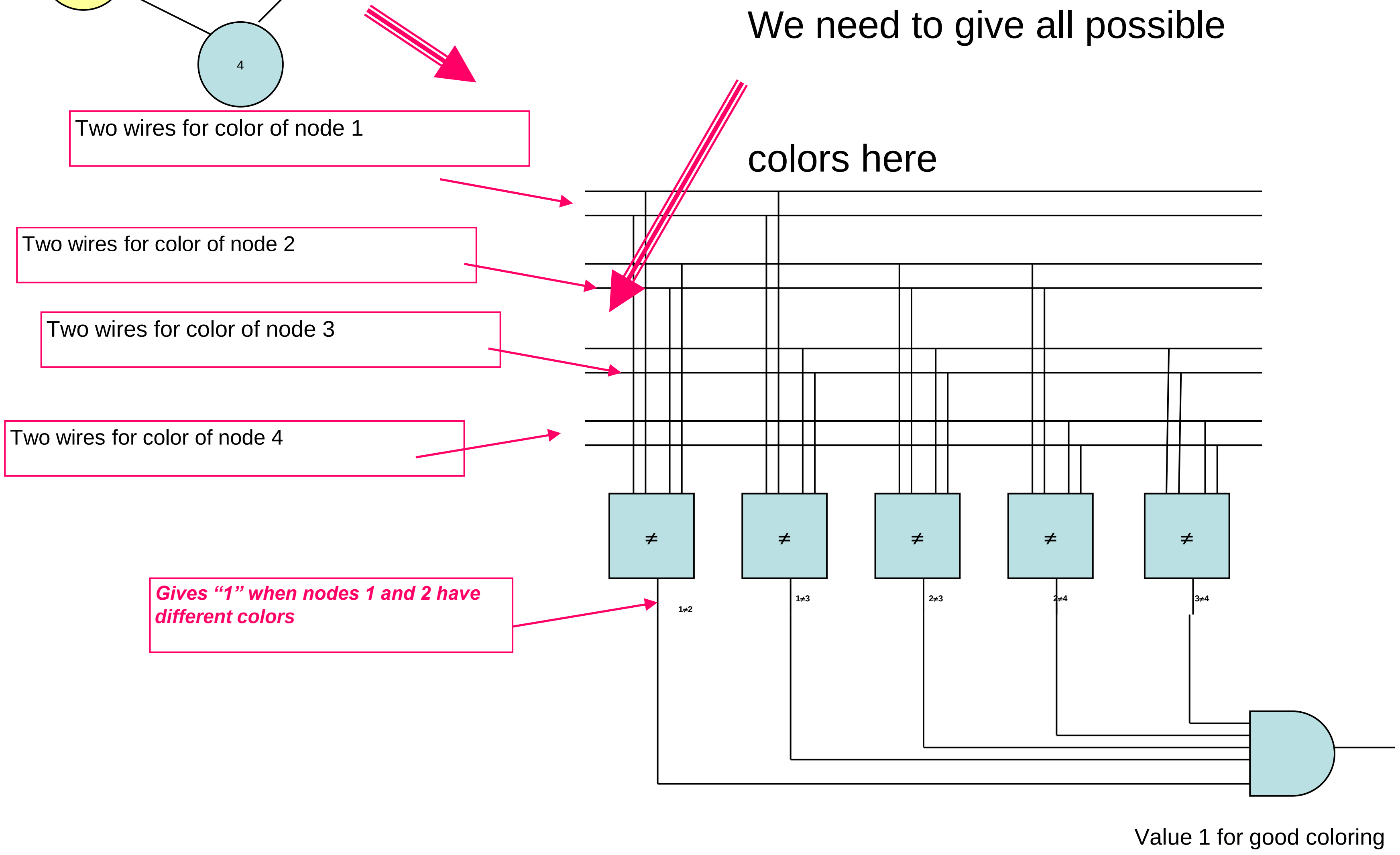
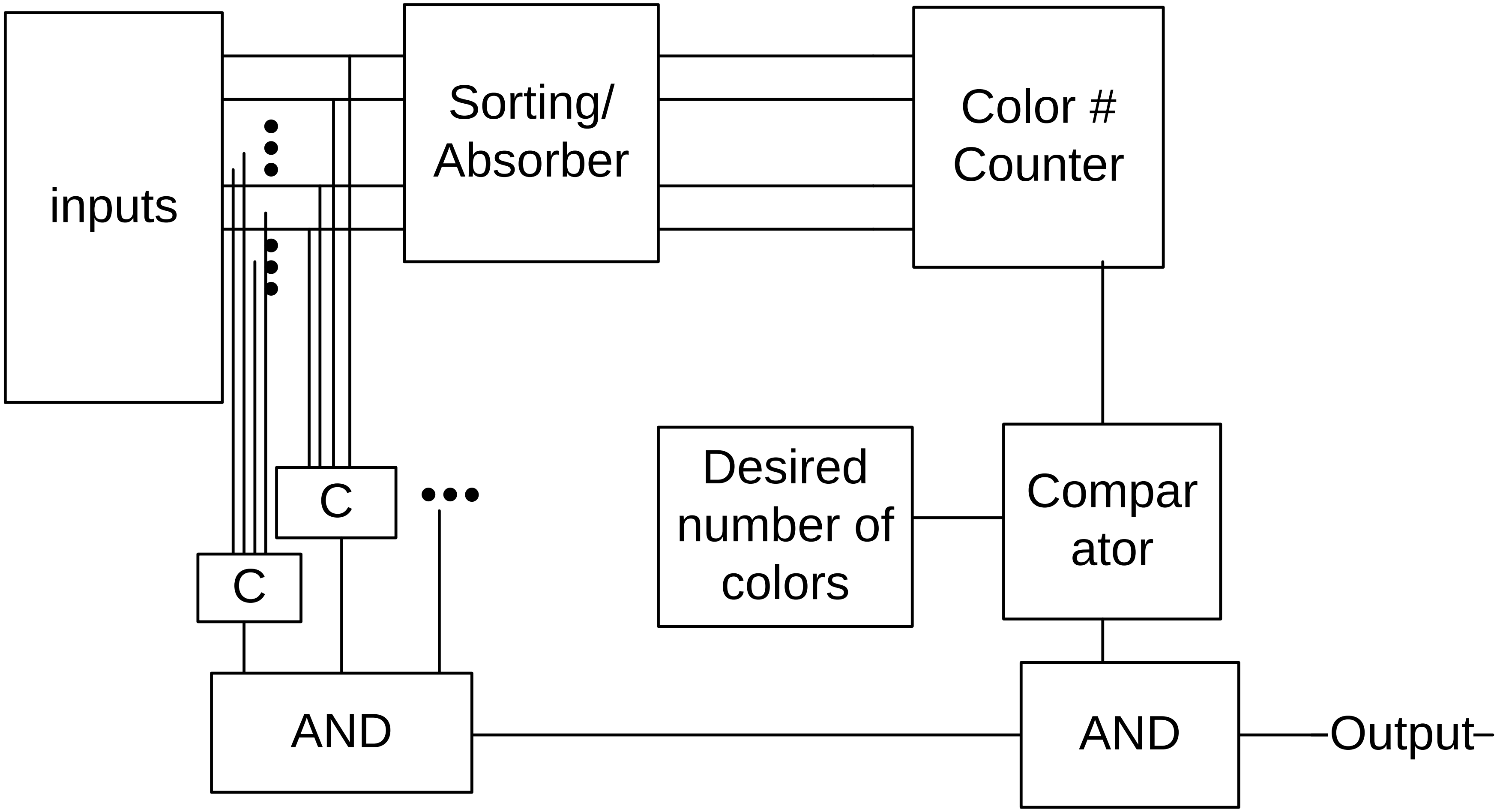

1
2
3
4
We need to give all possible colors here
Two wires for color of node 1
Two wires for color of node 2
Two wires for color of node 3
Two wires for color of node 4
≠
≠
≠
≠
≠
Gives "1" when nodes 1 and 2 have different colors
1≠2
1≠3
2≠3
2≠4
3≠4
Value 1 for good coloring

Figure 6a: A simple graph coloring problem

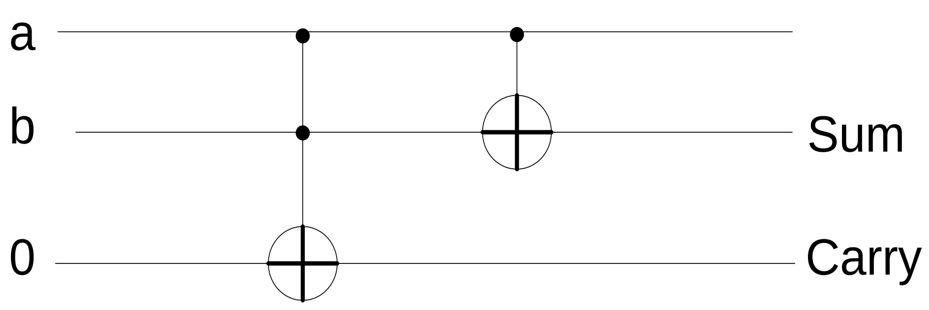


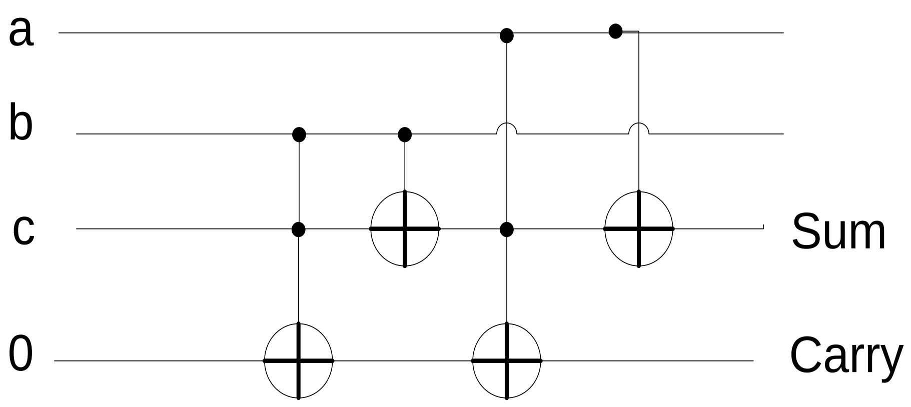

We need to give all possible colors here
Two wires for color of node 1
Two wires for color of node 2
Two wires for color of node 3
Two wires for color of node 4
≠
≠
≠
≠
≠
Gives "1" when nodes 1 and 2 have different colors
1≠2
1≠3
2≠3
2≠4
3≠4
Value 1 for good coloring

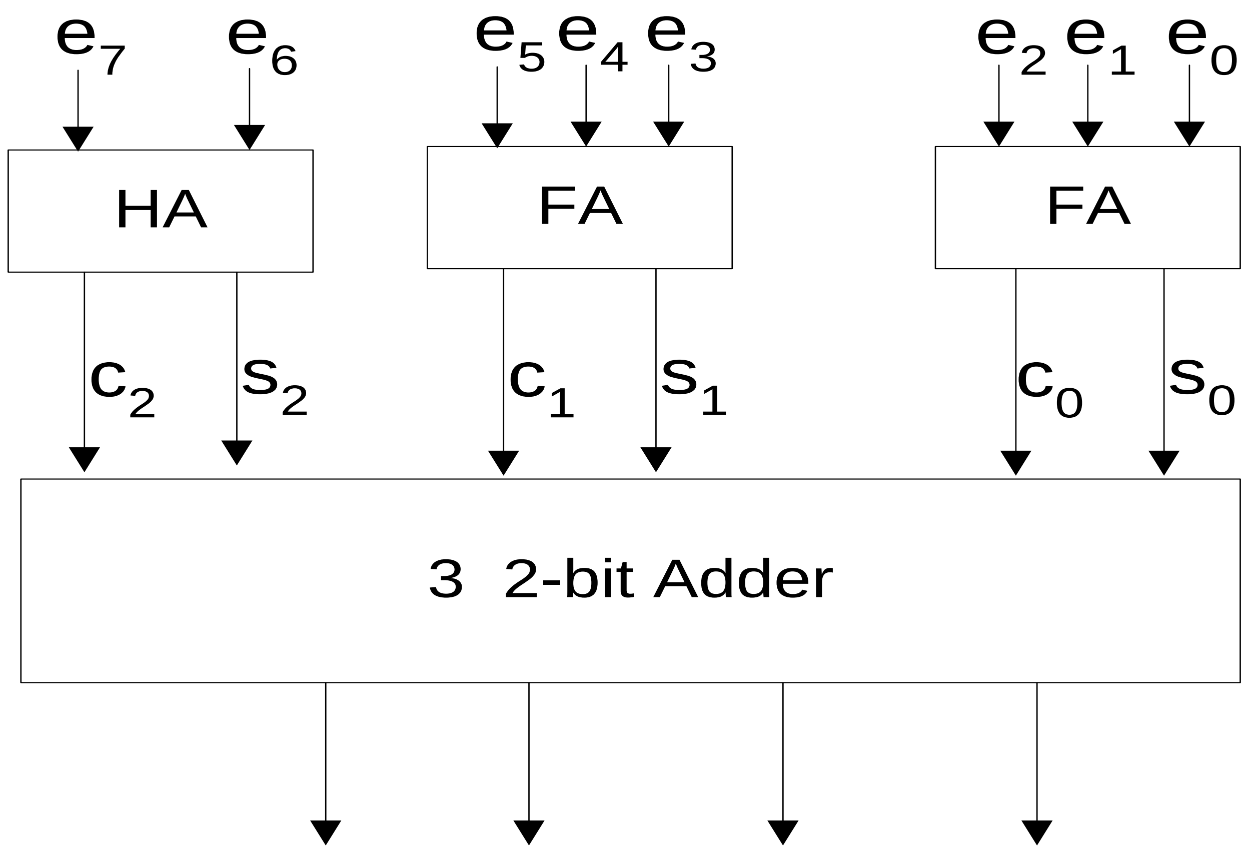
Simplified schematic of our Graph Coloring Oracle.



Quantum Adders.
(a)Half-adder realized using Toffoli and Feynman gates,
(b) (b) Full adder realized using two Toffoli, Feynman and inverter gates.







Block "Count Ones" realized using binary Half-Adders and Full-Adders.

The block at the bottom is the adder from Figure 9.2.

BINARY EQUALITY COMPARATOR

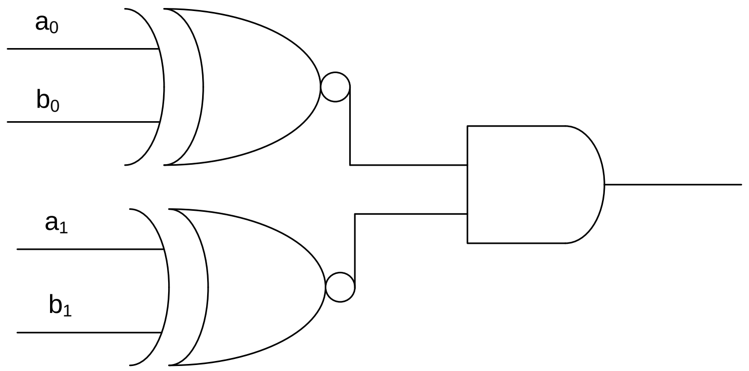
$$\bar{M} = \bar{a}_0\bar{a}_1\bar{b}_0\bar{b}_1 + \bar{a}_0a_1\bar{b}_0b_1 + a_0\bar{a}_1b_0\bar{b}_1 + a_0a_1b_0b_1$$

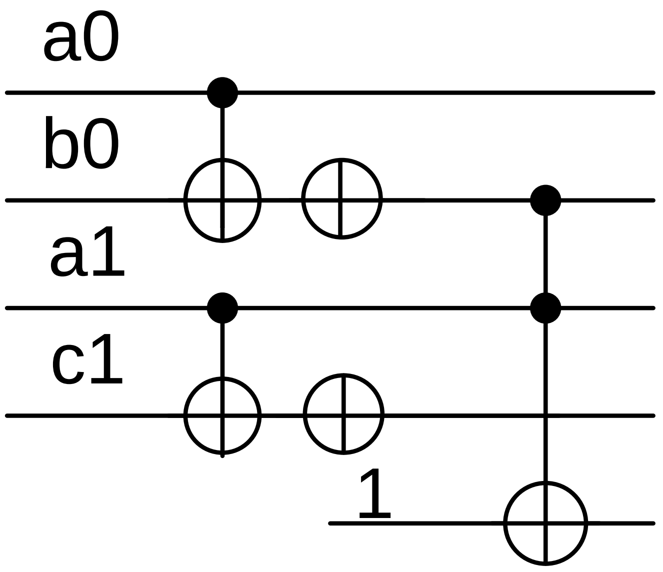
$$= \bar{a}_0\bar{b}_0(\bar{a}_1\bar{b}_1 + a_1b_1) + a_0b_0(\bar{a}_1\bar{b}_1 + a_1b_1)$$

$$= (\bar{a}_1\bar{b}_1 + a_1b_1)(\bar{a}_0\bar{b}_0 + a_0b_0)$$

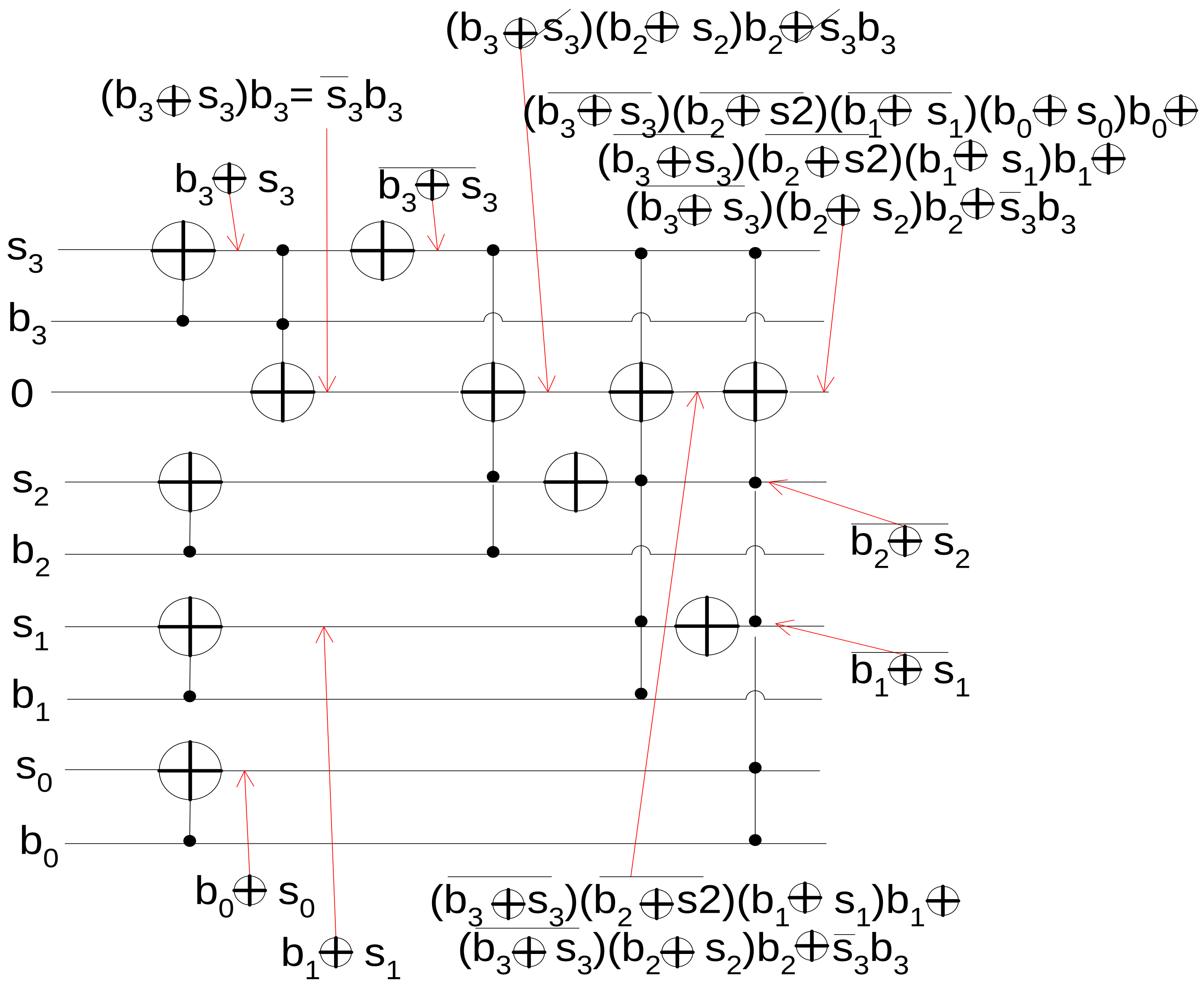
a_0a_1 \ b_0b_1	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

Inverted Karnaugh map of C block.

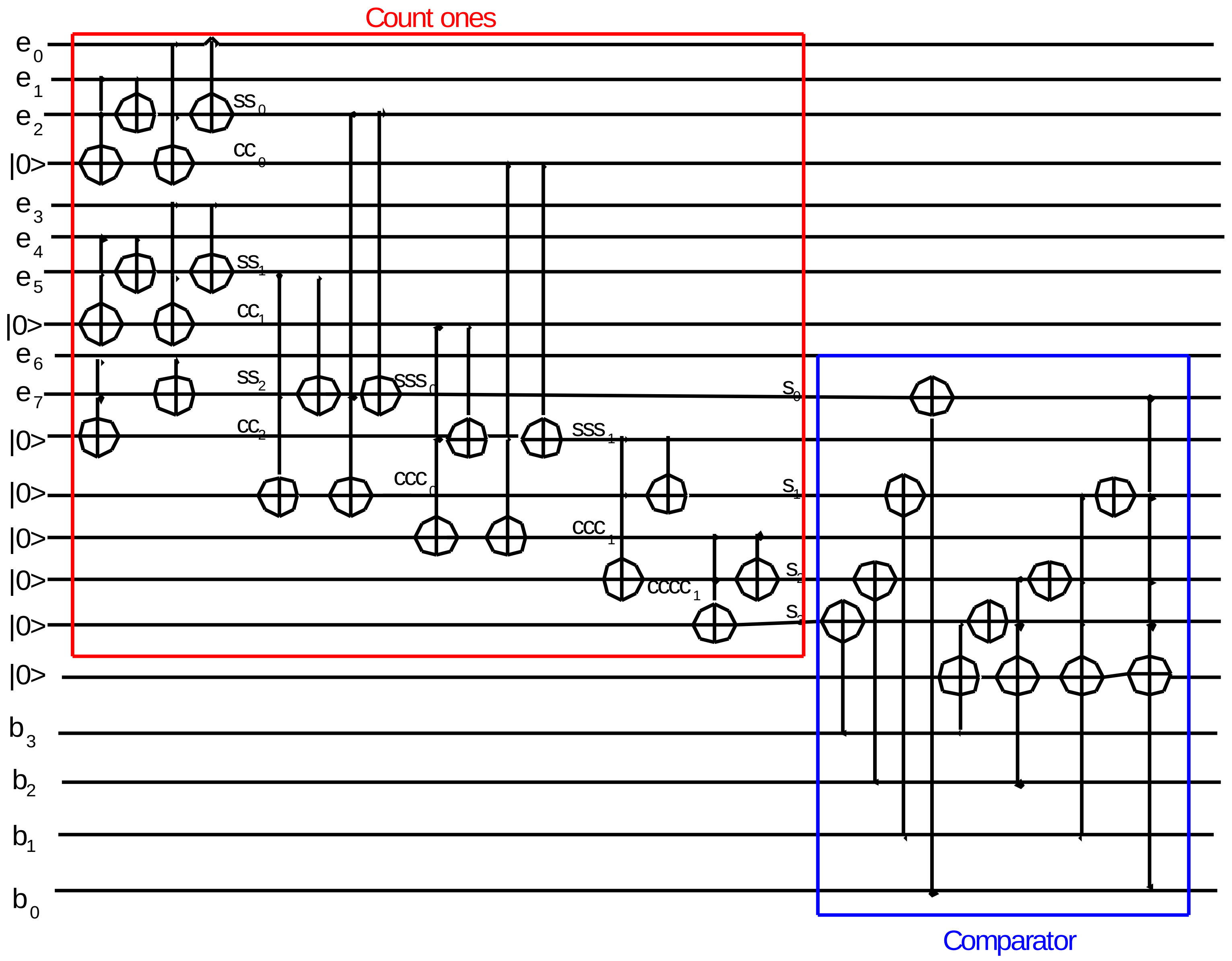




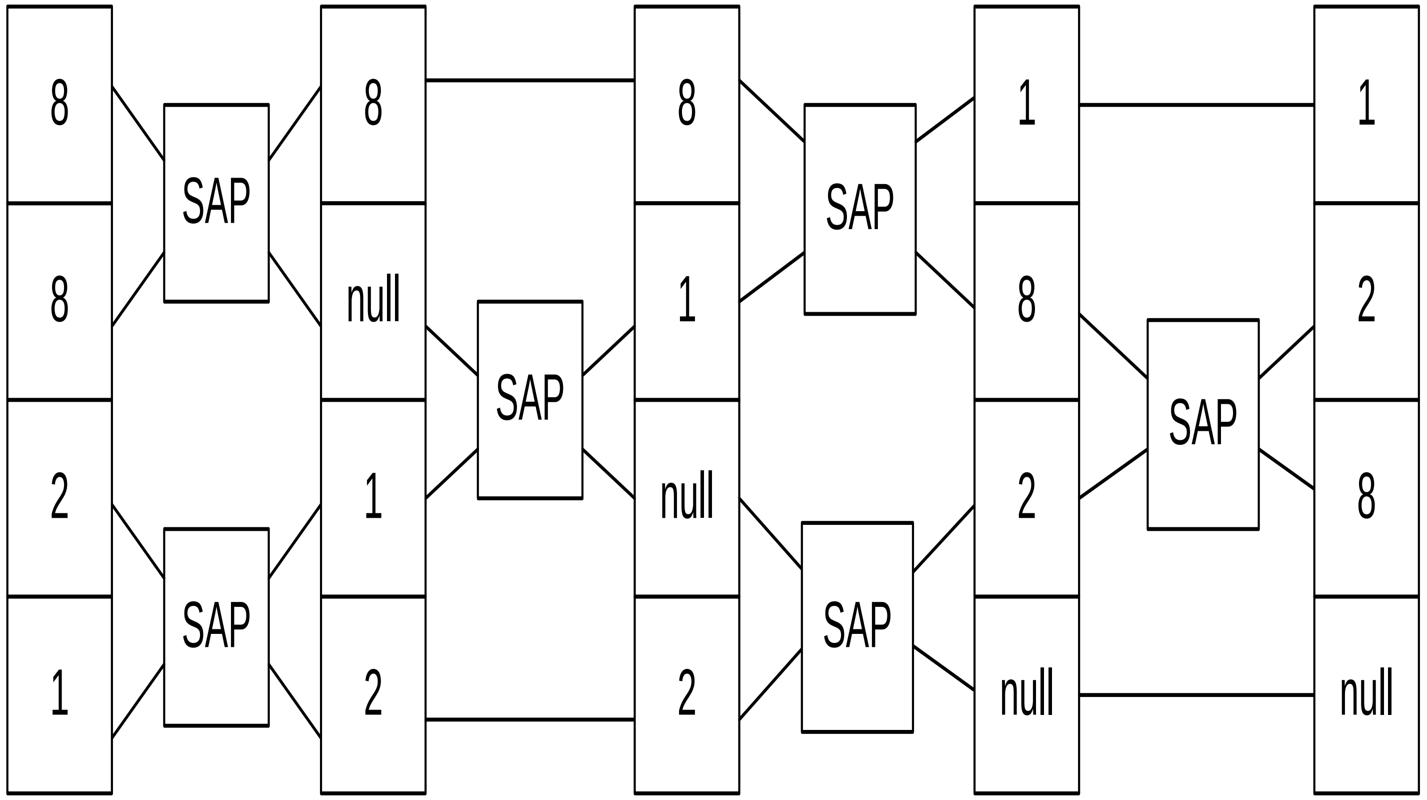
Quantum C Block


$(b_3 \oplus s_3)(b_2 \oplus s_2)b_2 \oplus s_3 b_3$
$(b_3 \oplus s_3)b_3 = \overline{s}_3 b_3$
$(\overline{b_3 \oplus s_3})(\overline{b_2 \oplus s2})(\overline{b_1 \oplus s_1})(b_0 \oplus s_0)b_0 \oplus$
$(\overline{b_3 \oplus s_3})(\overline{b_2 \oplus s2})(b_1 \oplus s_1)b_1 \oplus$
$(b_3 \oplus s_3)(b_2 \oplus s_2)b_2 \oplus \overline{s}_3 b_3$
$b_3 \oplus s_3$
$\overline{b_3 \oplus s_3}$
s_3
b_3
0
s_2
b_2
s_1
b_1
s_0
b_0
$\overline{b_2 \oplus s_2}$
$\overline{b_1 \oplus s_1}$
$b_0 \oplus s_0$
$b_1 \oplus s_1$
$(\overline{b_3 \oplus s_3})(\overline{b_2 \oplus s2})(b_1 \oplus s_1)b_1 \oplus$
$(\overline{b_3 \oplus s_3})(b_2 \oplus s_2)b_2 \oplus \overline{s}_3 b_3$

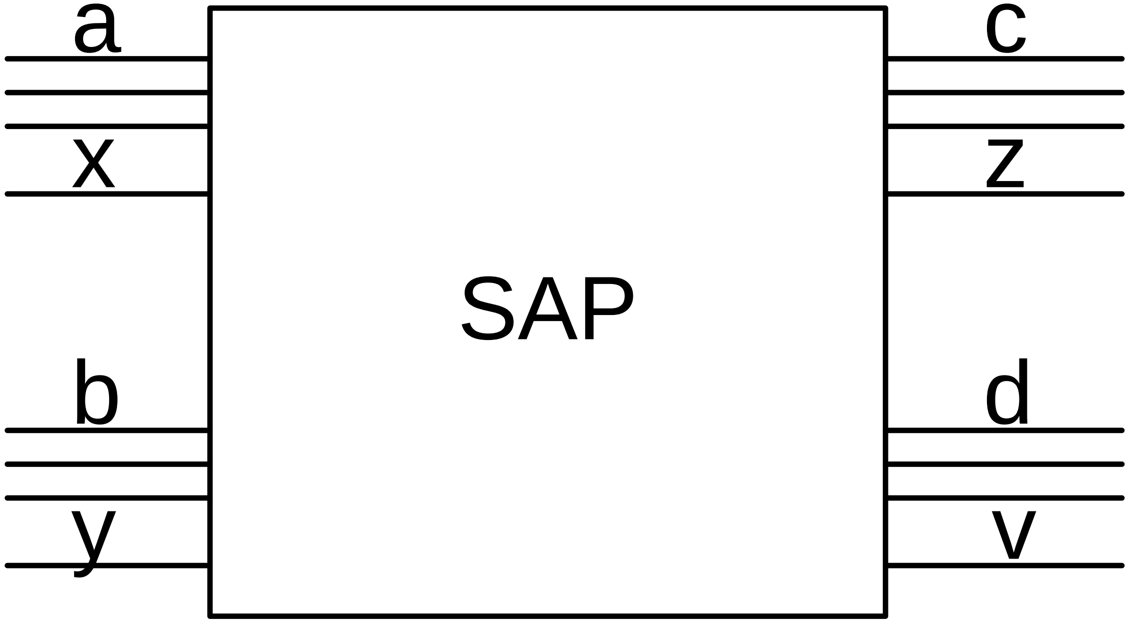
Binary Implementation of Comparator. Please observe the Toffoli gate with 5 inputs in AND.



- *Quantum Implementation of 8:4 Compressor and Comparator*



Butterfly iterative circuit for sorting/absorbing



SAP block

xy	a=b	a>b	a<b

00	c=null
	d=null
	z=0
	v=0
		c=null
		d=null
		z=0
		v=0
			c=null
			d=null
			z=0
			v=0

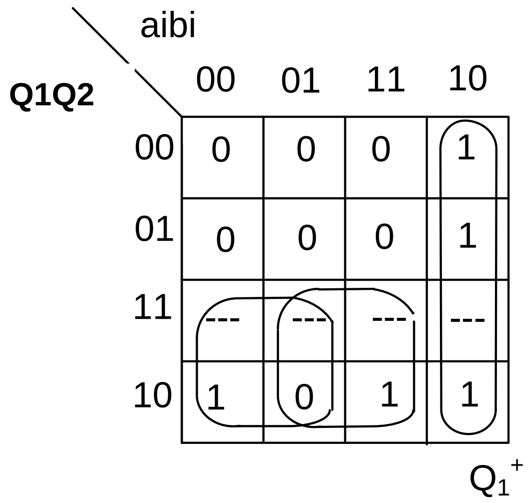
01	c=b
	d=null
	z=1
	v=0
		c=b
		d=null
		z=1
		v=0
			c=b
			d=null
			z=1
			v=0

11	c=a
	d=null
	z=1
	v=0
		c=a
		d=b
		z=1
		v=1
			c=b
			d=a
			z=1
			v=1

10	c=a
	d=null
	z=1
	v=0
		c=a
		d=null
		z=1
		v=0
			c=a
			d=null
			z=1
			v=0

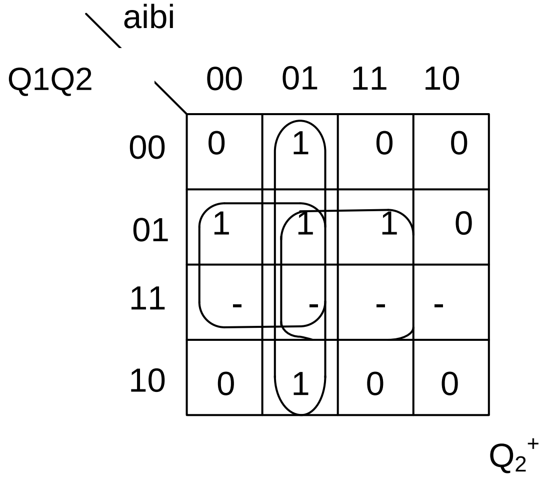
c,d,z,v

The map for c, d, z, and v signals



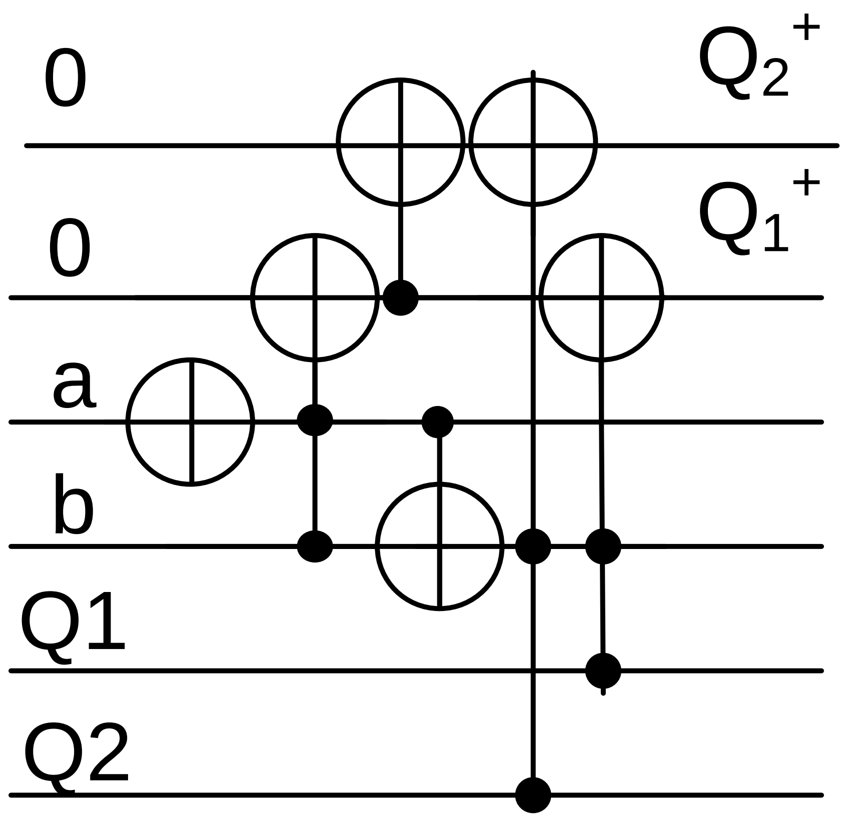
Q1Q2 \ aibi	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	---	---	---	---
10	1	0	1	1

Karnaugh map for Q_1+



Q1Q2 \ aibi	00	01	11	10
00	0	1	0	0
01	1	1	1	0
11	-	-	-	-
10	0	1	0	0

Karnaugh map for Q_2+



Circuit for Q_1+Q_2+