PROJECT 4:

Nikolay Nikolov

Problem 1:

Design a combinational circuit C1 with the following requirements.

The combinational logic C1 should contain at least 8 gates which include at least 3 OR and 3 NAND gates. Make sure that your design be different from those used by other students.

If your design is identical to the one used by other students, further investigation will be conducted and you will be asked to revise your design and redo the work.

Task 1:

1) Draw your circuit

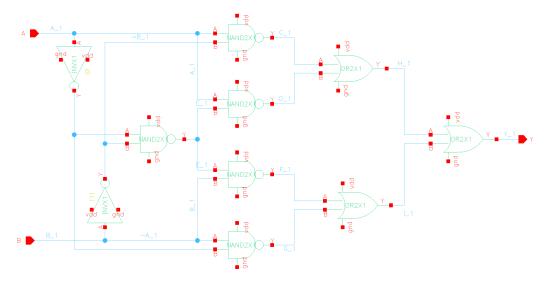


Fig. 1: Circuit Diagram for C1

2) You make an identical copy of C1 and name it as C2. Except the inputs, add the index 1 to all the wire names of C1, while adding the index 2 to all the wire names of C2.

Assume that the corresponding inputs use the same variable names, respectively.

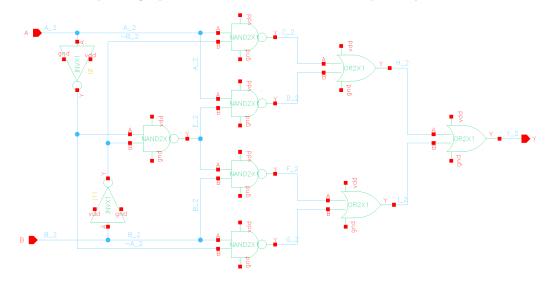


Fig. 2: C2 diagram

3) Download a SAT solver for satisfiability.

4) Prove the equivalence of C1 and C2 by a SAT solver. c Nikolay Nikolov c Mini-SAT ECE582 1 2 p cnf 10 8 -1 -2 5 0 4 2 -1 7 05 $2 \ 5 \ 6 \ 0$ 1 5 4 0 $1 -2 \ 3 \ 0$ 9 3 4 8 0 10 6 7 9 0 11 8 9 10 0 12 WARNING: for repeatability, setting FPU to use double precision 14 Problem Statistics 15 Number of variables: 16 10 17 Number of clauses: 18 Parse time: 0.00 s19 Eliminated clauses: $0.00~\mathrm{Mb}$ Simplification time: $0.00 \, s$ 20 21 22 Search Statistics ORIGINAL LEARNT 23 Conflicts | | Progress | Vars Clauses Literals | Limit Clauses Lit/Cl 2425 26 27 restarts conflicts: 0 28 (0 / sec)29 decisions : 1 (0.00 % random) (1230 / sec)30 (0 / sec)propagations : 0 31 conflict literals : 0 (-nan % deleted) 32Memory used : 12.00 MB33 CPU time : 0.000813 s34 SATISFIABLE 3536 SAT -1 -2 -3 4 5 -6 7 8 9 -10 0

Task 2:

Get a circuit C3 from C2 by replacing one gate in C2 with a gate of different functionality. Prove or disprove C1=C3 by a SAT solver.

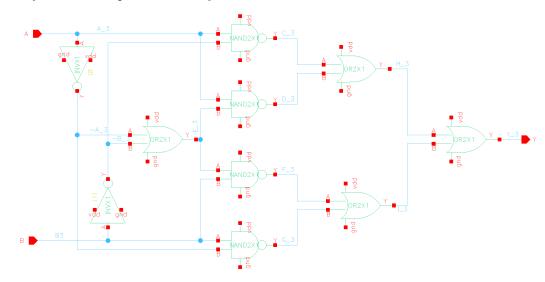


Fig. 3: C3 Circuit diagram, changing NAND gate at E_3 with OR gate

```
1
   WARNING: for repeatability, setting FPU to use double precision
                                 = Problem Statistics
3
      Number of variables:
                                         10
       Number of clauses:
       Parse time:
                                      0.00 s
       Eliminated clauses:
                                      0.00 \text{ Mb}
       Simplification time:
                                      0.00 s
9
                              Search Statistics
                            ORIGINAL
                                                         LEARNT
      Conflicts |
        | Progress |
                            Clauses Literals |
12
                                                   Limit Clauses Lit/Cl
```

```
13
14
15
    restarts
                              : 1
16
    conflicts
                              : 0
                                                  (-nan / sec)
                                                   (0.00 \% \text{ random}) (inf / sec)
17
    decisions
                              : 1
                                                  (-nan / sec)
    propagations
                              : 0
18
                                                  (-nan % deleted)
19
    conflict literals
                              : 0
20
    Memory used
                              : 12.00 MB
21
    \ensuremath{\mathrm{CPU}}\ \ensuremath{\mathrm{time}}
                              : 0 s
22
23
   SATISFIABLE
24
25 \quad \mathrm{SAT} \ -1 \ -2 \ -3 \ 4 \ 5 \ -6 \ 7 \ 8 \ 9 \ -10 \ 0
        All circuits have the same SAT. Hence they are equivalent. Adding the SAT
    in the input also gave a satisfiable solution.
    c Nikolay Nikolov
    c Mini-SAT ECE582
 3
   p cnf 10 8
    1 \ 2 \ 5 \ 0
 4
    2 -1 7 0
    2 \ 5 \ 6 \ 0
 6
    1 \ 5 \ 4 \ 0
8
    1 - 2 \ 3 \ 0
9
    3 4 8 0
    6 7 9 0
10
11 8 9 10 0
    -1 -2 -3 4 5 -6 7 8 9 -10 0
13
    nik@nik~/Downloads/minisat/build $ ./minisat-simp minisat.in
         minisat.out
    WARNING: for repeatability, setting FPU to use double precision
15
16
                               Problem Statistics
17
    WARNING! DIMACS header mismatch: wrong number of clauses.
       Number of variables:
20
       Number of clauses:
                                           0.00 s
       Parse time:
21
        Eliminated clauses:
22
                                           0.00 \text{ Mb}
23
        Simplification time:
                                           0.00 \, \mathrm{s}
24
                               Search Statistics
    | Conflicts |
                               ORIGINAL
                                                                LEARNT
```

Progress

```
27
                       Vars Clauses Literals | Limit Clauses Lit/Cl
28
29
30
    restarts
31
    conflicts
                             : 0
                                                 (-nan / sec)
                                                 (0.00 \% \text{ random}) (\text{inf /sec})
32
    decisions
                             : 1
33
    propagations
                             : 0
                                                 (-nan / sec)
                                                 (-nan % deleted)
34
    conflict literals
                             : 0
35
                             : 12.00 MB
    Memory used
    CPU time
                             : 0 s
37
   SATISFIABLE
38
39
   SAT
40
    -1 2 3 4 5 -6 -7 8 9 -10 0
42
    \label{eq:Warning:formula} W\!A\!R\!N\!I\!N\!G\!\colon \mbox{for repeatability} \;,\;\; \mbox{setting FPU to use double precision}
43
44
                                    Problem Statistics
45
   WARNING! DIMACS header mismatch: wrong number of clauses.
46
    Number of variables:
                                            10
48
       Number of clauses:
                                            10
49
       Parse time:
                                          0.00 s
       Eliminated clauses:
                                          0.00 \text{ Mb}
50
                                          0.00 s
       Simplification time:
52
                                Search Statistics
53
      Conflicts |
                              ORIGINAL
                                                              LEARNT
         Progress
55
                 Vars Clauses Literals |
                                                      Limit Clauses Lit/Cl
56
57
58
   restarts
                             : 1
59
   conflicts
                             : 0
                                                 (0 /sec)
   decisions
                                                 (0.00 \% \text{ random}) (674 / \text{sec})
                             : 1
                             : 0
61
    propagations
                                                 (0 / sec)
    conflict literals
                             : 0
                                                 (-nan % deleted)
                             : 12.00 \text{ MB}
63
    Memory used
64
   CPU time
                             : 0.001483 s
65
```

```
66 SATISFIABLE
67
68 SAT
69 -1 -2 -3 4 5 -6 7 8 9 -10 0
```

Problem 2.

Equivalence checking by implicit state enumeration.

Consider the product machine with shared input P=S1xS2 in your project 2. Following the lecture on implicit state enumeration, use the implicit state enumeration to check the equivalence of S1 and S2.

The initial states of FFs in both circuits are all the equivalent states. Use the Boolean algebra to do the reasoning.

Implicit State Enumeration

Reach-ability Analysis

Fig. 4: Reachability

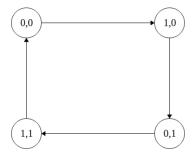
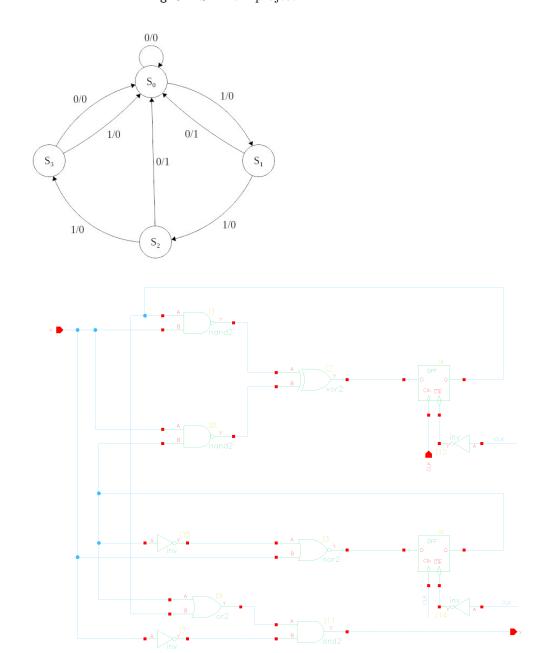


Fig. 5: State Diagram

Fig. 6: FSM from project 2



Equivalence Checking

```
S0 = v1v2
R1 = av1v1' + av1v1' + av1v1' + av1v1'
R2 = av2v2' + av2v2' + av2v2' + av2v2
\exists a\exists v1\exists v2.S0 \land R1 \land R2
\exists a\exists v1\exists v2.(\neg v1\neg v2) \land (av1v1' + av1v1' + av1v1' + av1v1') \land (av2v2' + av2v2' + av2v2' + av2v2')
= \exists a\exists v1\exists v2.(\neg av1v\neg 2v1'\neg v2' + a\neg v1\neg v2\neg v1'v2')
```