CENTER OF NANOELECTRONICS & DEVICES

CND 121:VLSI

Project : Automatic Transistor Sizing for combinational CMOS logic circuits

Section : 17

Group Name: group_17_121_6

Submitted by:

Student Name	ID
Fagr Ahmed Abdelrazek	v23010586
Nada Walid Radwan	v23010647
Mohamed Yasser	v23010495

Submitted to : Dr Mohamed Saleh

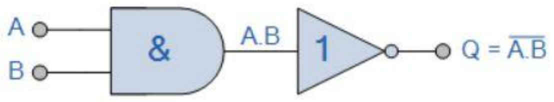
Automatic Transistor Sizing for combinational CMOS logic circuits

Introduction:

Optimal gate sizing of transistors is amongst the fundamental problems in VLSI circuit designing. Size of transistors in any CMOS logic circuit directly affects its performance in terms of speed, power dissipation etc. An efficient digital CMOS integrated circuit (IC) presents an optimal trade-off between speed and power dissipation. For circuits which constitute a small number of transistors, manual transistor sizing is possible. But the ever increasing demand for portable electronic devices with enhanced features has triggered the design of digital CMOS circuits with very high complexity. As a result, the task of designing and optimizing these circuits for target specifications has become extremely cumbersome. This necessitates inclusion of automation in the design and optimization flow of CMOS logic circuits such that the least amount of manual effort is involved in optimizing smaller logic blocks with varying specifications for design reuse in high complexity IC design.

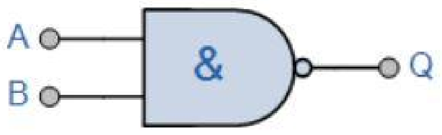
NAND GATE:

The NAND (Not – AND) gate has an output that is normally at logic level "1" and only goes "LOW" to logic level "0" when **ALL** of its inputs are at logic level "1". The **Logic NAND Gate** is the reverse or "*Complementary*" form of the AND gate we have seen previously.



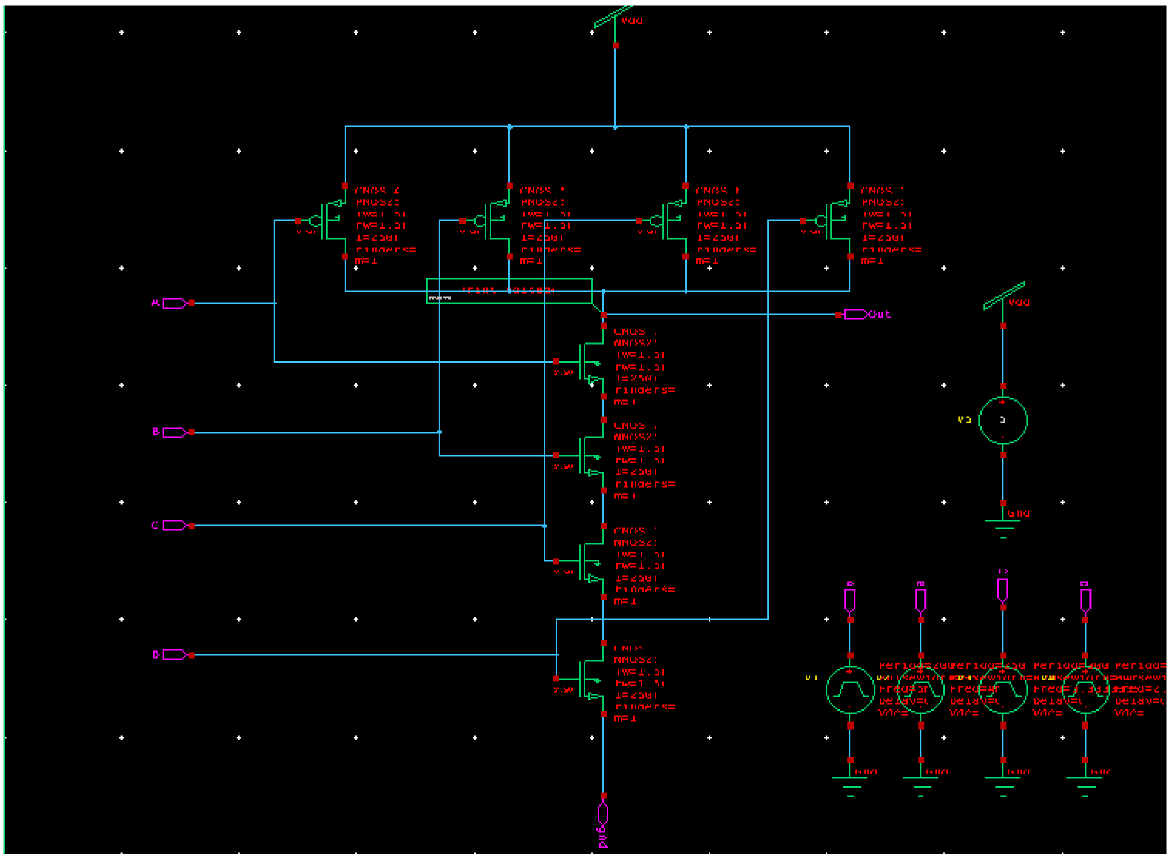
2-input "AND" gate plus a "NOT" gate

The logic or Boolean expression given for a logic NAND gate is that for *Logical Addition*, which is the opposite to the AND gate, and which it performs on the *complements* of the inputs. Its Boolean expression is denoted by a single dot or full stop symbol, (.) with a line or *Overline*, (‾) over the expression to signify the NOT or logical negation of the NAND gate giving us the Boolean expression of: A.B = Q.

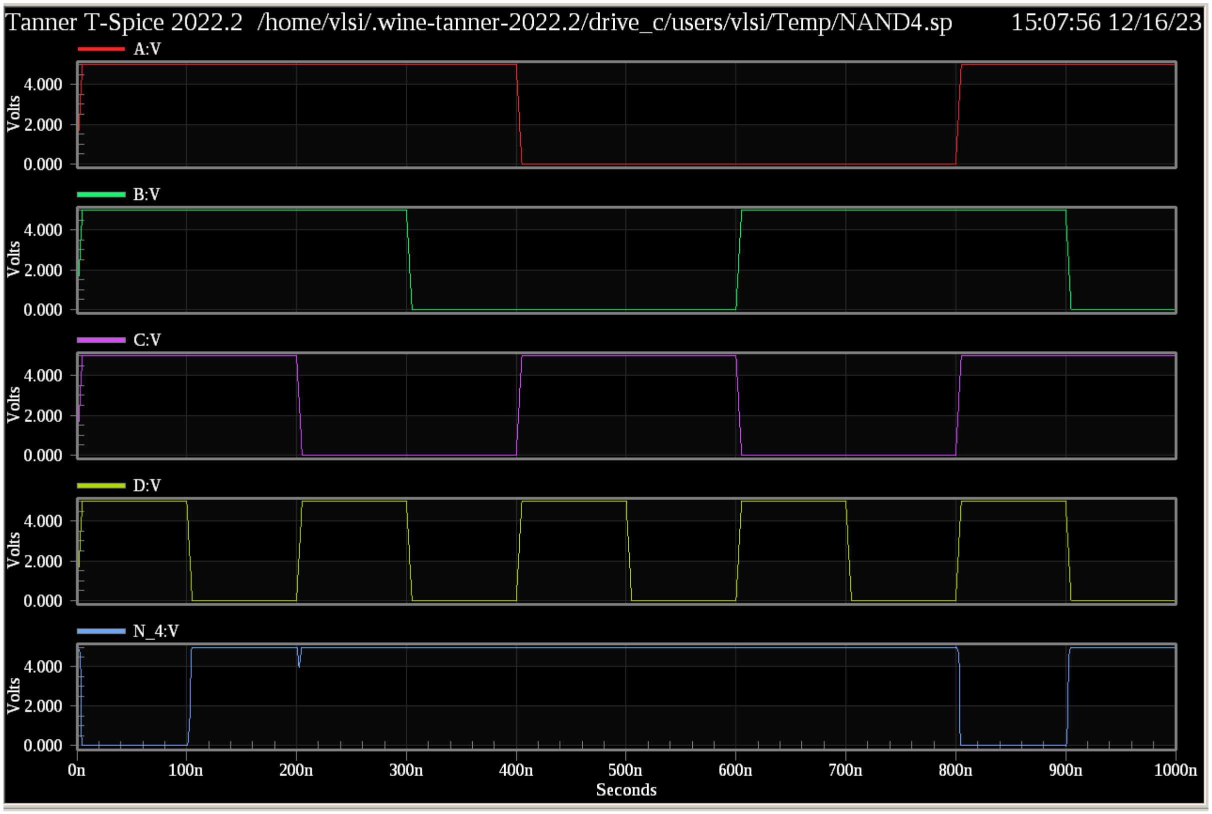
Symbol	Truth Table		
	B	A	Q
 2-input NAND Gate	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression **Q = $\overline{A.B}$**	Read as A **AND** B gives **NOT** Q		

1. NAND 4 Inputs:

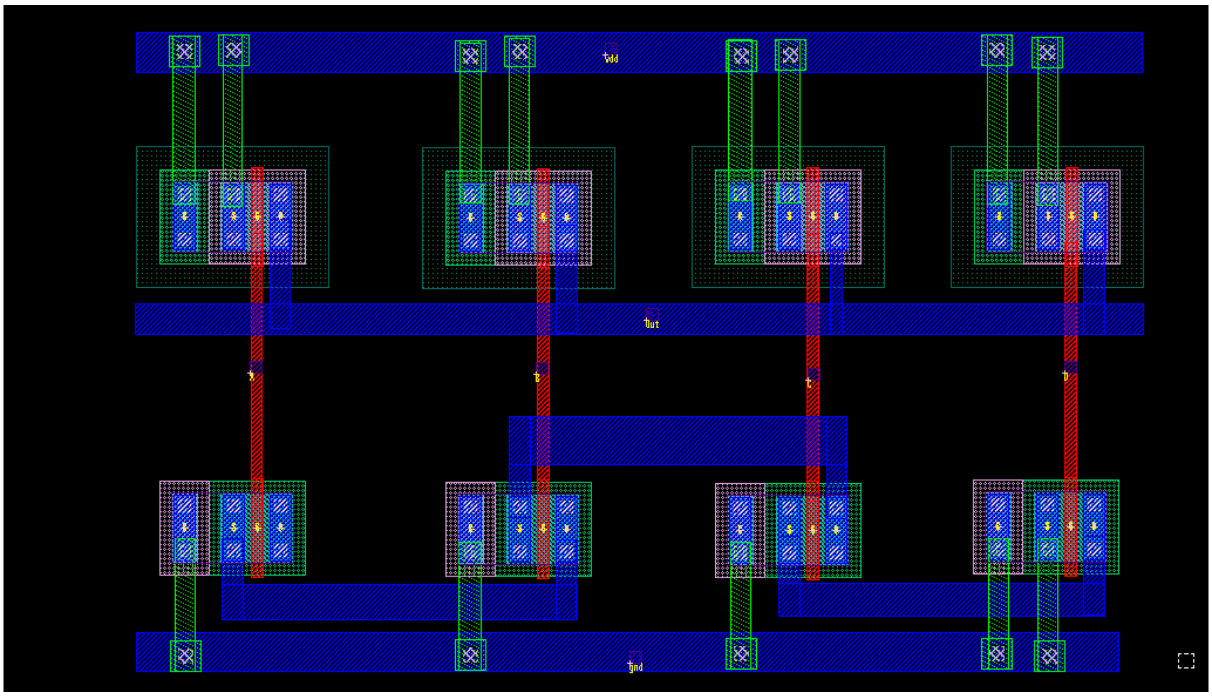
a. Circuit Simulator



b. Wave


Tanner T-Spice 2022.2 /home/vlsi/.wine-tanner-2022.2/drive_c/users/vlsi/Temp/NAND4.sp 15:07:56 12/16/23

c. Layout Editor



d. DRC

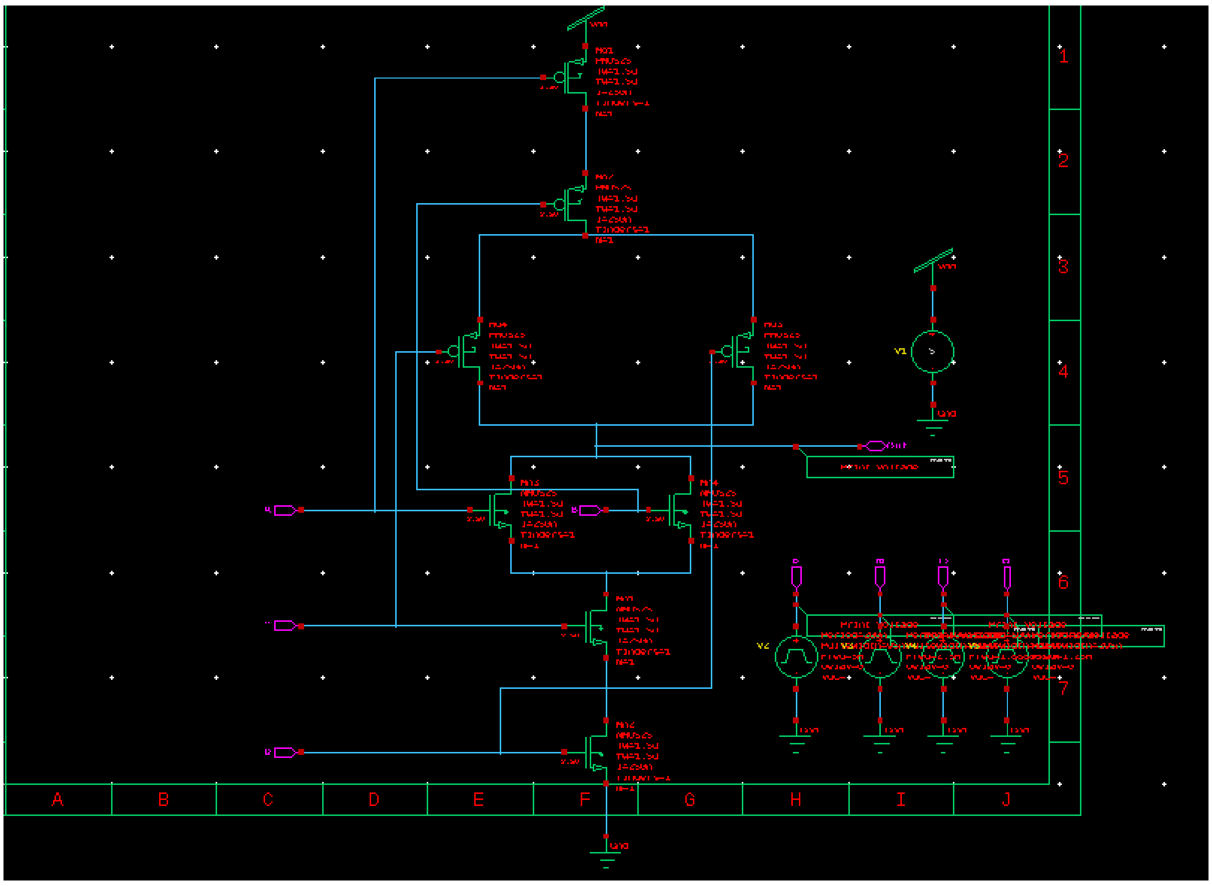
```
--------------------------------------------------------------------------------
--- RULECHECK RESULTS STATISTICS (BY CELL)
---
--------------------------------------------------------------------------------
--- SUMMARY
---
TOTAL CPU Time:                  0
TOTAL REAL Time:                 0
TOTAL Original Layer Geometries: 110 (221)
TOTAL DRC RuleChecks Executed:   97
TOTAL DRC Results Generated:     0 (0)
```

e. LVS

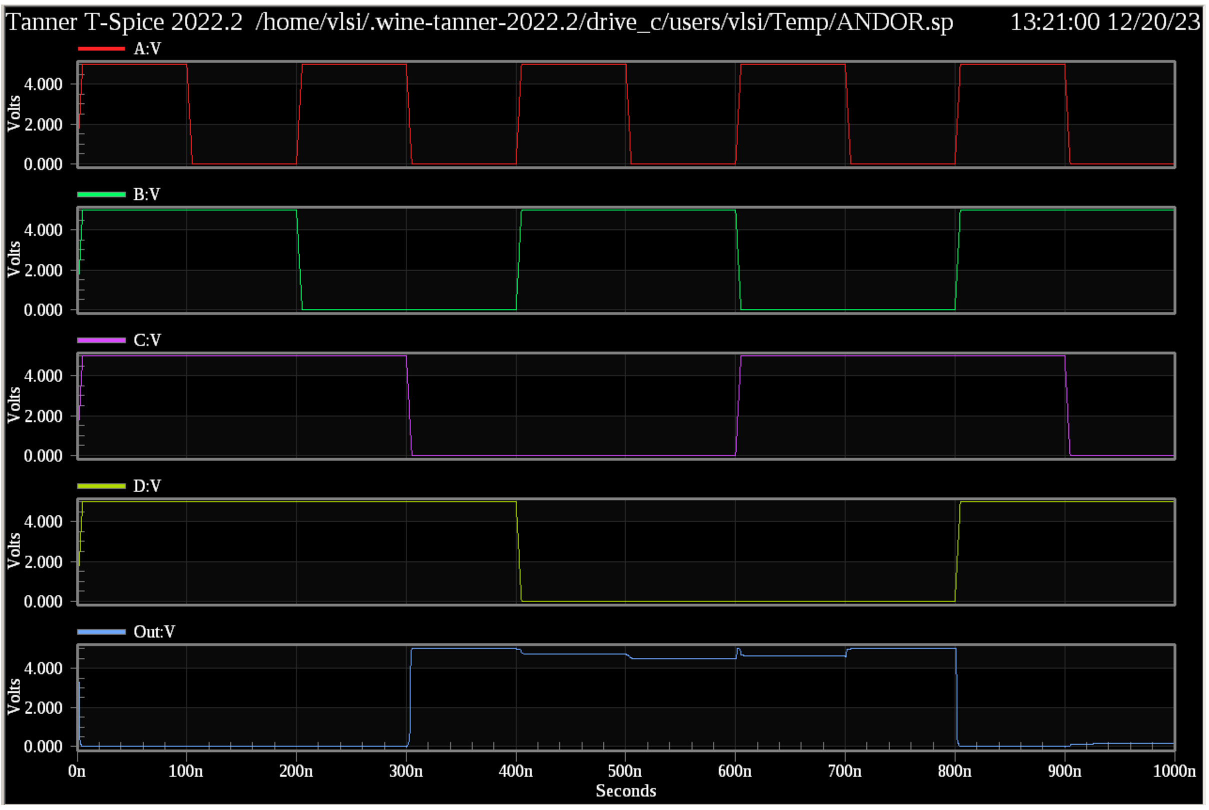
```
                   ##################################################
                   ##                                              ##
                   ##           C A L I B R E    S Y S T E M       ##
                   ##                                              ##
                   ##               L V S    R E P O R T           ##
                   ##                                              ##
                   ##################################################


REPORT FILE NAME:          NAND4.lvs.report
LAYOUT NAME:               NAND4.sp ('NAND4')
SOURCE NAME:               NAND4.src.net ('NAND4')
RULE FILE:                 _Generic_250nm_LVS.cal_
CREATION TIME:             Wed Dec 20 12:57:55 2023
CURRENT DIRECTORY:         /home/vlsi/Desktop/VLSIPROJECT
USER NAME:                 vlsi
CALIBRE VERSION:           v2023.3_29.15    Tue Aug 1 13:25:59 PDT 2023


                             OVERALL COMPARISON RESULTS



                         #       ###################       _   _
                        #        #                 #       *   *
                   #   #         #     CORRECT     #         |
                    # #          #                 #       \___/
                     #           ###################


**************************************************************************************************
                                         CELL  SUMMARY
**************************************************************************************************

  Result         Layout                         Source
  -----------    -----------                    --------------
  CORRECT        NAND4                          NAND4


**************************************************************************************************
                                         LVS PARAMETERS
**************************************************************************************************
```

2. (A+B)CD:

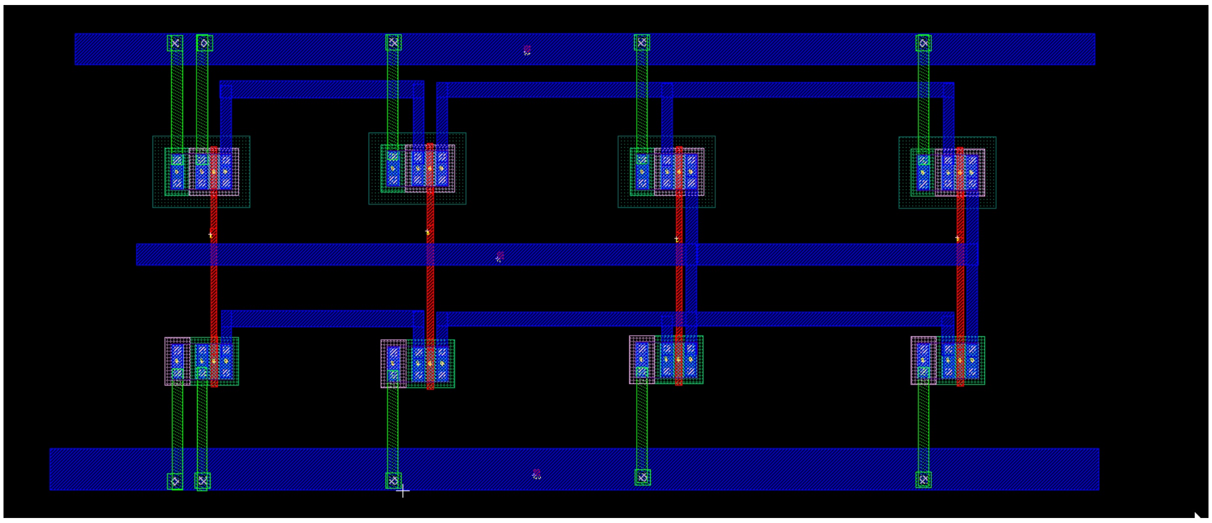
a. Circuit Simulator



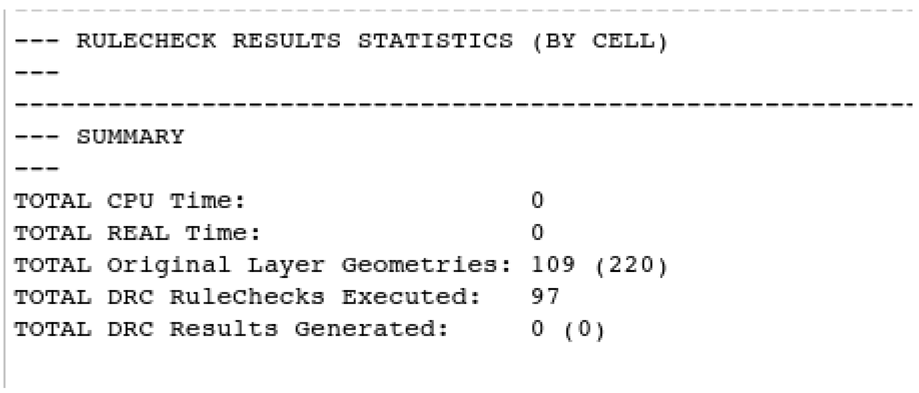
b. wave



c. Layout Editor

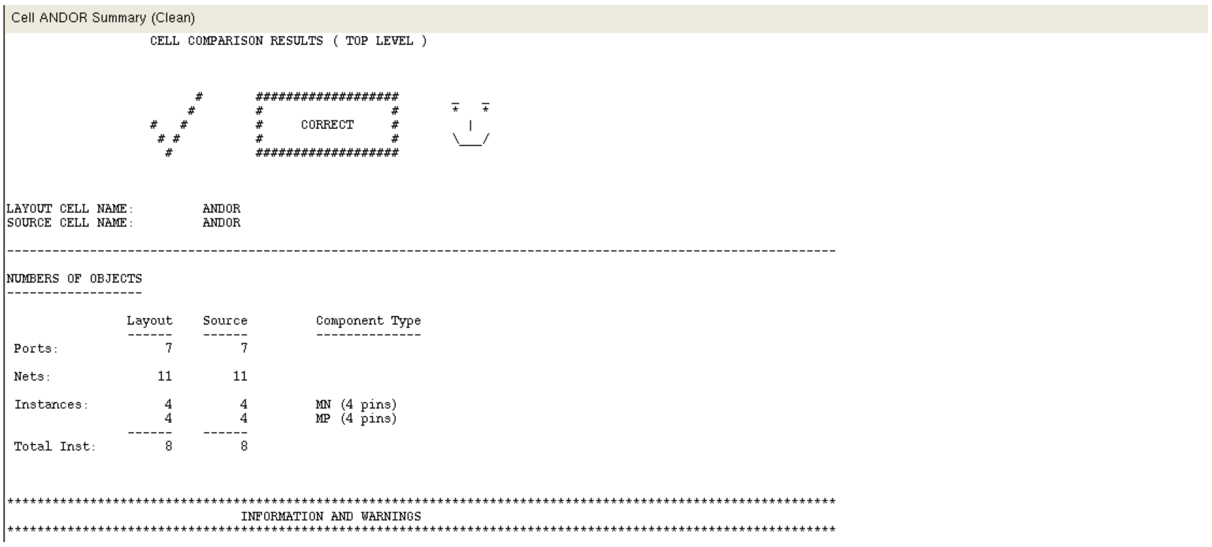


d. DRC



```
--- RULECHECK RESULTS STATISTICS (BY CELL)
---
---------------------------------------------------------
--- SUMMARY
---
TOTAL CPU Time:                  0
TOTAL REAL Time:                 0
TOTAL Original Layer Geometries: 109 (220)
TOTAL DRC RuleChecks Executed:   97
TOTAL DRC Results Generated:     0 (0)
```

e. LVS



Cell ANDOR Summary (Clean)

```
                   CELL COMPARISON RESULTS ( TOP LEVEL )



                        #      ###################
                       #       #                 #
                  #   #        #     CORRECT     #    _   _
                   # #         #                 #     |
                    #          ###################    \___/


LAYOUT CELL NAME:         ANDOR
SOURCE CELL NAME:         ANDOR

--------------------------------------------------------------------------------------------------------------

NUMBERS OF OBJECTS
------------------

                Layout    Source         Component Type
                ------    ------         --------------
 Ports:            7         7

 Nets:            11        11

 Instances:        4         4           MN (4 pins)
                   4         4           MP (4 pins)
                ------    ------
 Total Inst:       8         8


**************************************************************************************************************
                               INFORMATION AND WARNINGS
**************************************************************************************************************
```

3. Python General code:

```
import numpy as np
from scipy.optimize import minimize

def objective(x):      # Objective: minimize the overall delay while considering area and tpHL = tpLH

    return x[0] * (x[1] + x[2]) + x[3]

def constraint1(x):    # Constraint 1: Area Constraint

    return x[0] * (x[1] + x[2])

def constraint2(x):    # Constraint 2: tpHL = tpLH

    return x[1] * x[2] - x[3]

# initial guesses
n = 4
x0 = np.zeros(n)
x0[0] = 1.0  # Width of PMOS
x0[1] = 1.0  # Width of NMOS1
x0[2] = 1.0  # Width of NMOS2
x0[3] = 1.0  # Length of NMOS1 and NMOS2

# bounds for transistor sizes
bounds = [(1.0, 5.0), (0.1, 5.0), (0.1, 5.0), (0.1, 5.0)]

# show initial objective
print('Initial Objective: ' + str(objective(x0)))

# define constraints
con1 = {'type': 'ineq', 'fun': constraint1}
con2 = {'type': 'eq', 'fun': constraint2}
constraints = [con1, con2]

# optimize
solution = minimize(objective, x0, method='SLSQP', bounds=bounds, constraints=constraints)
x = solution.x

# show final objective
print('Final Objective: ' + str(objective(x)))

# print solution
print('Solution:')
print('Width of PMOS: ' + str(x[0]))
print('Width of NMOS1: ' + str(x[1]))
print('Width of NMOS2: ' + str(x[2]))
print('Length of NMOS1 and NMOS2: ' + str(x[3]))
```