****

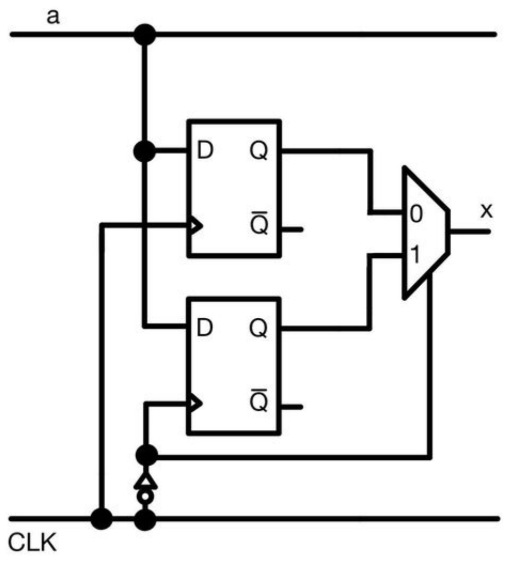
EE/CS 119A Homework 5

Edward Speer

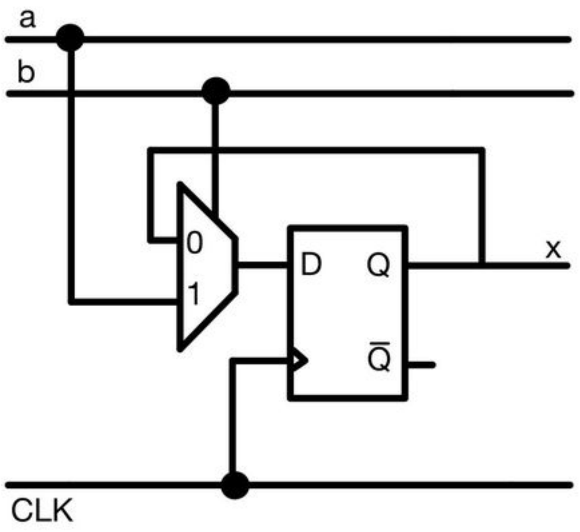
November 6, 2024

****

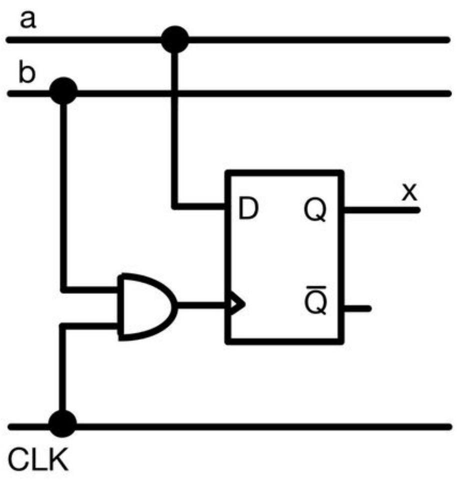
1. **Problem 1**:
   Since CLK is in the sensitivity list, we need to update x to reflect the value of a on both the rising and falling edges of the clock. We therefore need 2 DFFs to detect each edge and store the value of a at that time. We then know which DFF should be output by the value of the CLK signal. If the clock is high, we output the value of the DFF storing the value of a on the rising edge and vice versa.



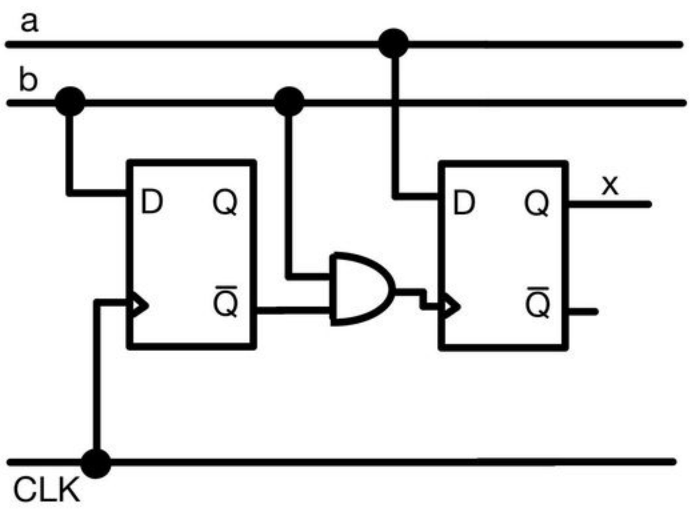
2. **Problem 2**:
   Since the clock is in the sensitivity list, but we only update x when the clock is high, we only need a single DFF to detect the rising edge of the clock. When we do detect the rising edge, we simply check if b is high, and either pass through a or keep x the same accordingly.



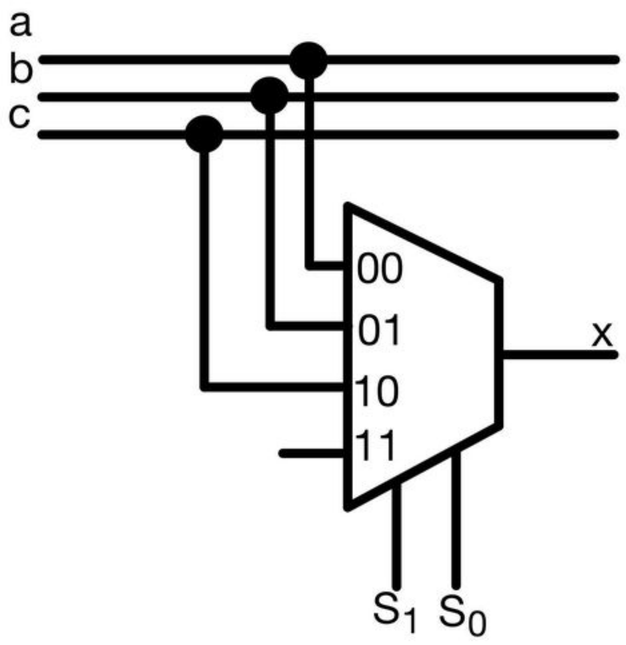
3. **Problem 3**:
   In this case, with both b and the clock in the sensitivity list, but only updating x when both are high, we need to detect any time the clock or b have a rising edge while the other is high. So feed the two signals into an AND gate and detect a rising edge on the AND gate's output. When either signal has a rising edge, if the other signal is high, the AND gate will have a rising edge.



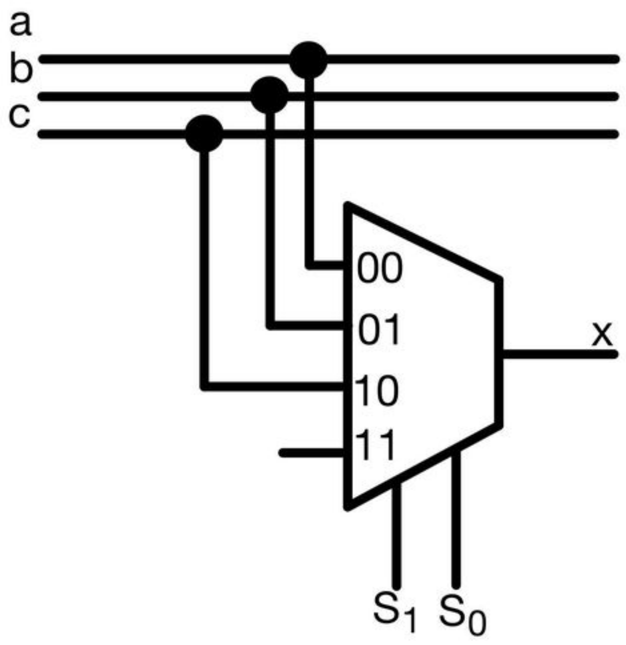
4. **Problem 4**:
   In this problem, only the clock is in the sensitivity list. We then check if the clock is high before continuing. Therefore, activity should only be triggered on the rising edge of the clock. However, we then demand a rising edge on b. This means that this circuit is looking for the extremely rare condition of simultaneous rising edges on both the clock and on b. We won't be able to ensure that the two rising edges are exactly simultaneous, but what we do do is detect whether or not b has experience a rising edge within the propagation delay time of a DFF. Begin by clocking a DFF on the rising edge of the clock. Feed b into the DFF. Feed the $\overline{Q}$ output of the DFF into an AND gate with b. This is essentially ANDing b with the value of $\overline{b}$. Now, if b has a rising edge within the propagation delay of the DFF, the $\overline{Q}$ output will be high, and b will go high, meaning there will be a very short pulse on the output of the AND gate. In only this case, we will update x to be the value of a. This is a horrible circuit that essentially depends on a glitch, and I don't recommend anyone ever build it.



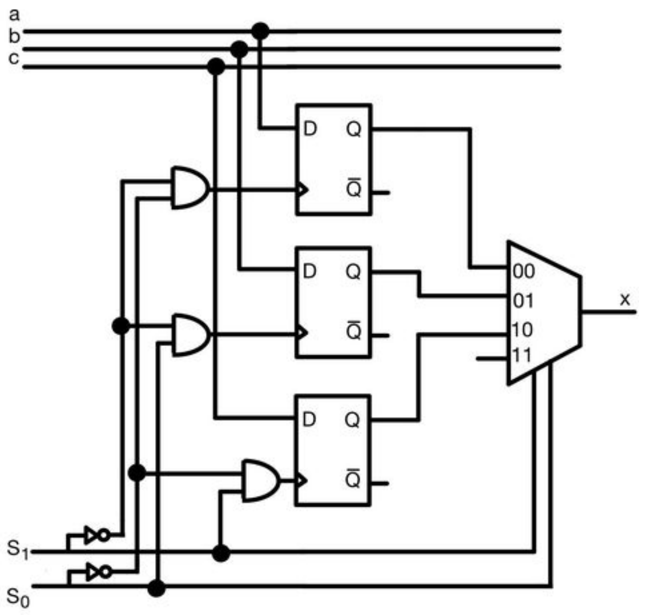
5. **Problem 5**:
   This is a simple mux. Outside of a process this is simply concurrent logic that udates whenever an input to the mux changes. Note that this will synthesize to a 4:2 mux, but there are only 3 inputs, so the 4th input is tied to 0.



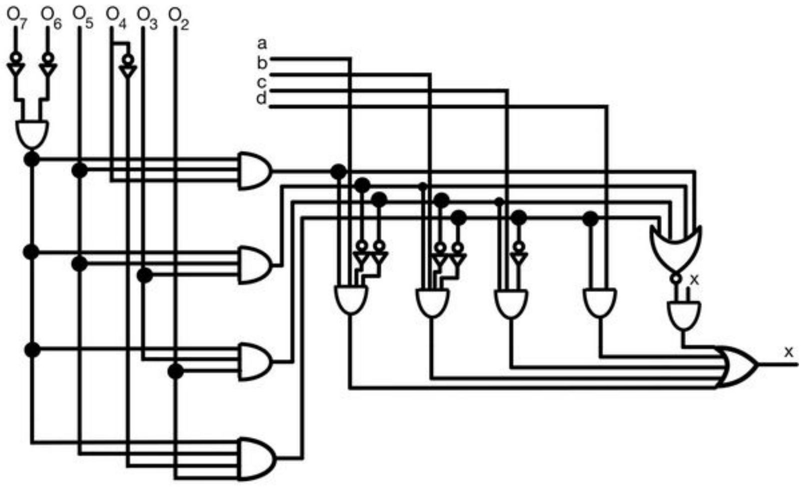
6. **Problem 6**:
   Luckily, every input to this mux is included in the sensitivity list, so that the circuit does not change from the previous problem. When any input to the mux changes, the output will update so that we essentially have concurrent logic. Notice again we must tie off one of the inputs to the mux to 0.



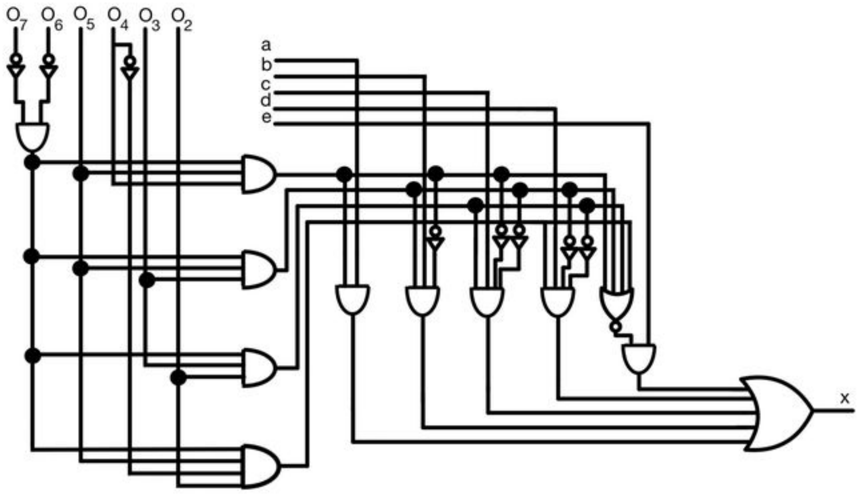
7. **Problem 7**:
Since now only the control signals are included in the sensitivity list, we need to essentially cache the values of the inputs to the mux according to when the control signals are updated. To do so, use 3 DFFs, one for each input to the mux. Clock each of the DFFs on an AND gate that will go high for the rising edge of the control signals that correspond to that input. Then, when the control signals are updated, the DFF will store the value of the input at that time, and pass it through accordingly. We once again have to tie off the 4th input to the mux to 0.



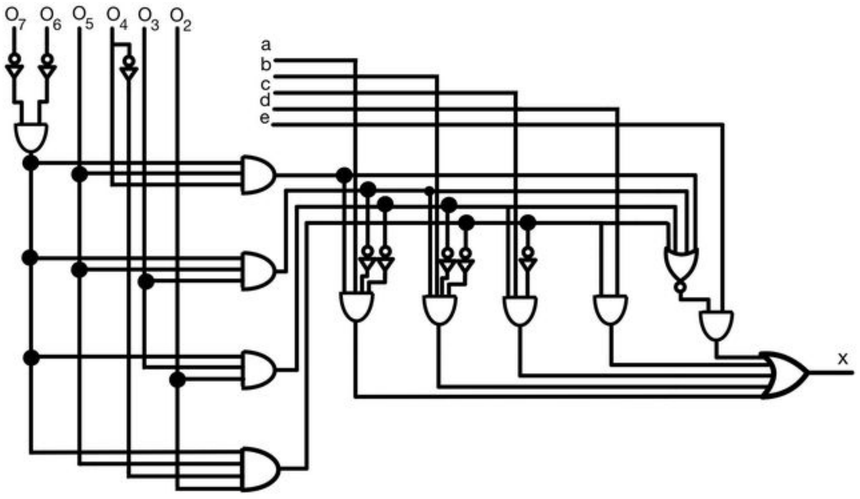
8. **Problem 8**:
Notice in the code that (a) x is a signal, and (b) instead of using if/elsif/else, each case is a separate if statement. As a result of (a), any change to x is scheduled at time $t + \Delta$, and as a result of (b), the if statement priorities will be last to first, meaning if the third to last if statement is true, it will overwrite the value of x set by the second to last or first if statement. Therefore, the circuit should essentially check each if condition. Then assign x according to the first condition AND the exclusion of all other conditions, OR the second condition AND the exclusion of all later conditions, etc. There is no default assignement for x, so if none of the conditions are met, x will not be updated.



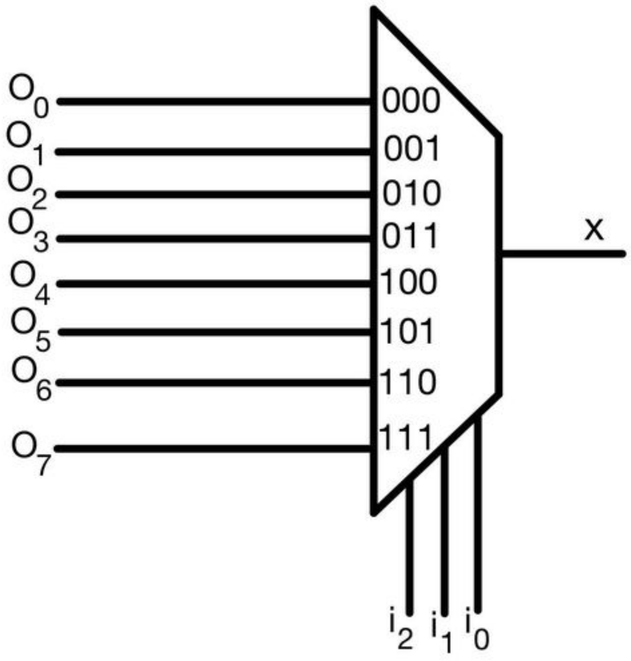
9. **Problem 9**:
   In this case, there are two notable changes. Firstly, we are now using if/elsif/else. This switches the order of precedence so that if the first condition is true, we will not check any later conditions so that earlier conditions supercede later ones. Secondly if none of the conditions are true x will take on e as a default now instead of not updating.



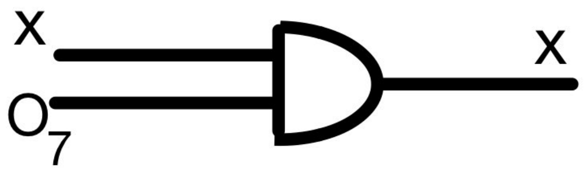
10. **Problem 10**:
    In this case, we have reverted to using if/then statements alone, reversing the order of priority back to the last cases taking precedence over earlier ones. Notice that before the if statements are evaluated, x is set to e. However, remember that since x is a signal, this assignment is scheduled for time $t+\Delta$. This means this assignment will be overwritten by any if condition that is true. Therefore, this assignment to e will function like a default if none of the if conditions are met.



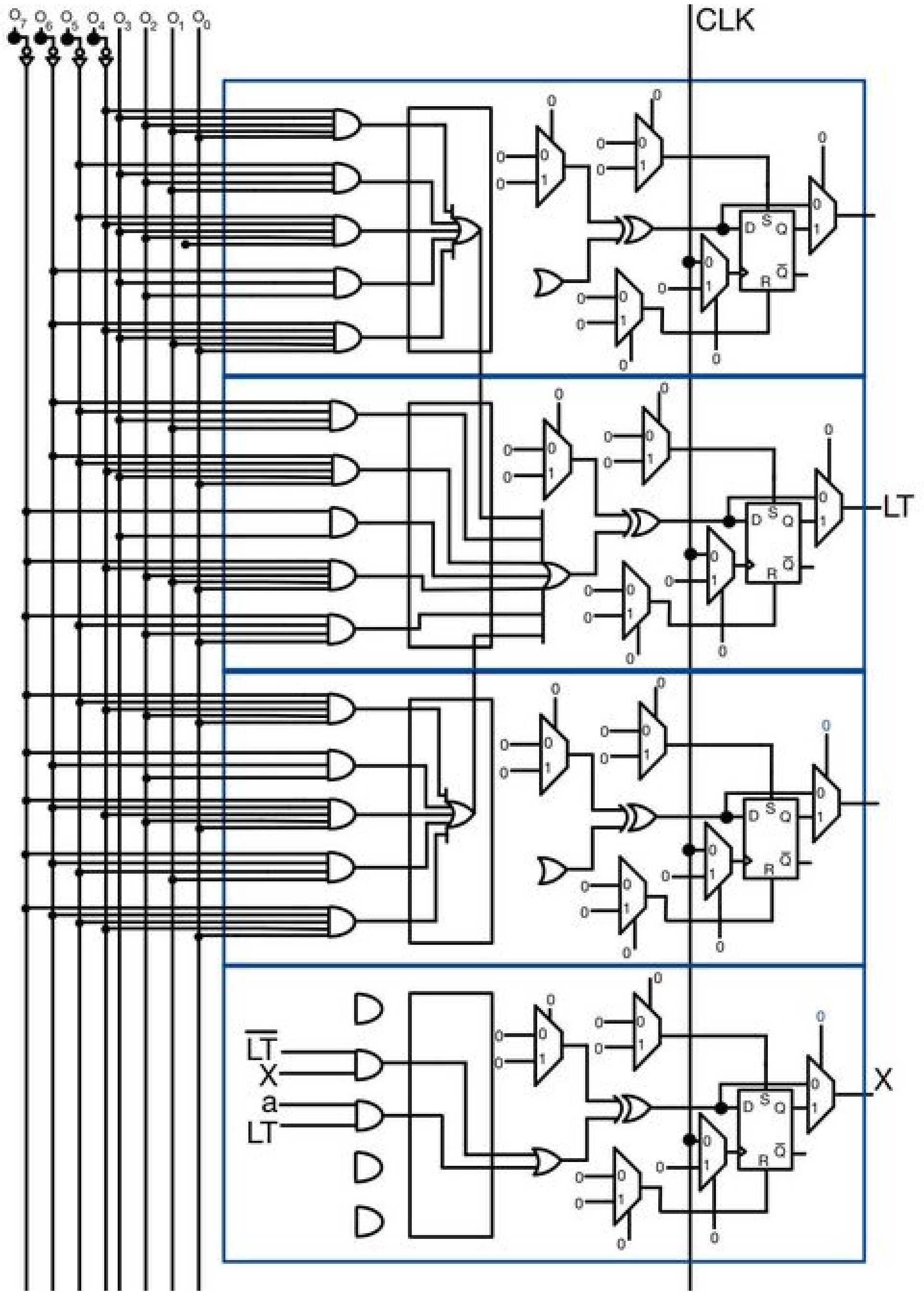
11. **Problem 11**:
This is a simple 8:3 mux. Since i is an integer ranging up to 7 it will take 3 bits to represent all 8 values. We will simply pass each bit of the 8 bit opcode into a different input of the mux and set the control signals to the 3 bit value of i so that the mux will output the value of the bit corresponding to i.



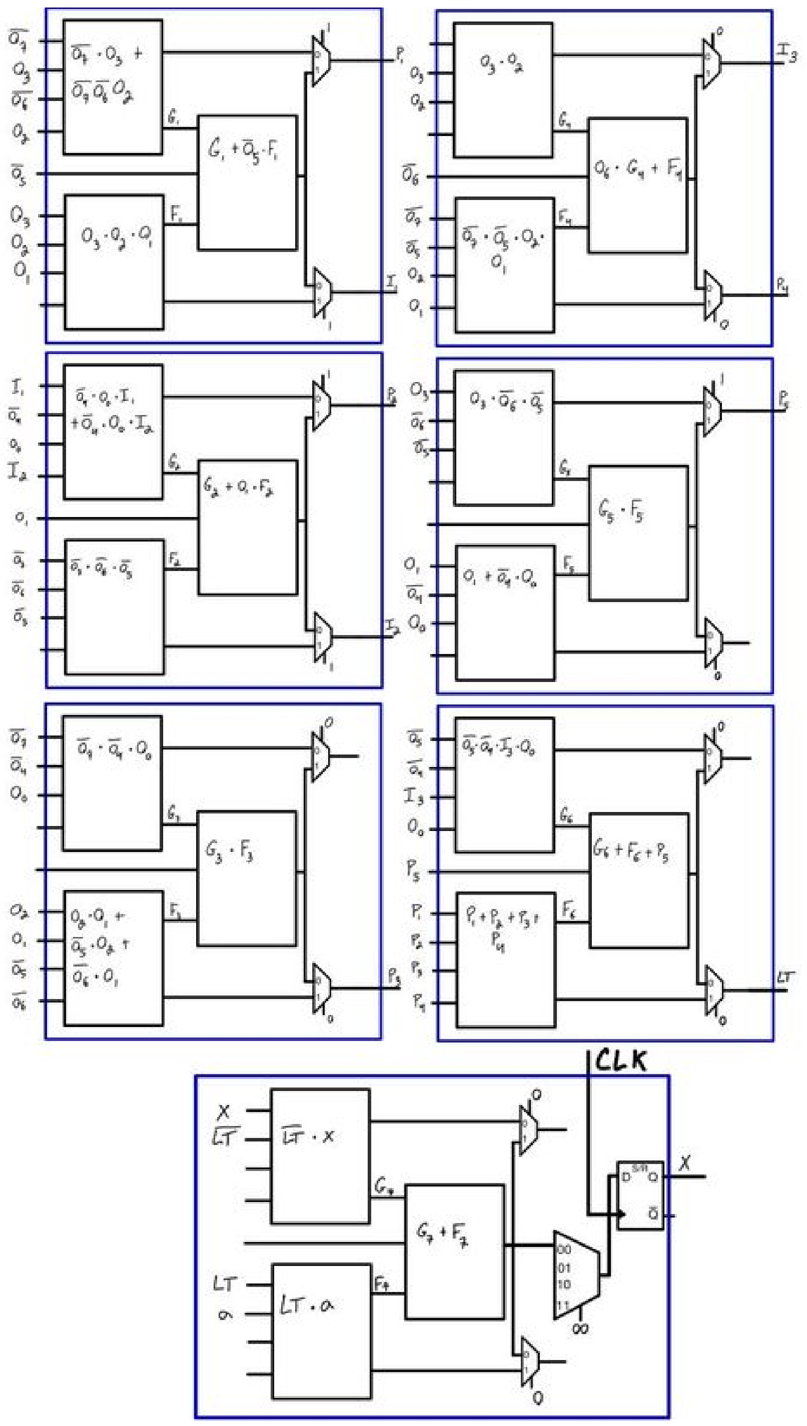
12. **Problem 12**:
For this problem, recall that x is a signal. This means that any changes to the value of x are scheduled for time $t + \Delta$. So for this code, setting x to a is scheduled but not executed. This assignment is then overwritten by each successive assignment to x. Thus, no execution of the loop will have any impact on x except the final iteration. Therefore we will very simply set x to x AND opcode[7]. Since the opcode is in the sensitivity list, and changes in MSB of the opcode are the only thing that can possibly change the value of x we don't need to worry about timing or clocking.



13. **Problem 13**:
For this problem, we need to build a comparator that checks if the upper nibble of opcode is less than the lower nibble. Then, we use the output of the comparator which I named LT. If LT is high, we set x to a, otherwise we set x to x. To simplify the logic of the comparator, I used the Quine-McCluskey python package I developed for homework 3. The full code of the package and the script which uses it to generate the prime implicants chart is attached. The prime implicants chart for the comparator is also attached. I then implemented the strict SOP form from the prime implicants chart in the given archtecture resulting in the following.



14. **Problem 14**:
Use the same comparator logic from the previous problem. I therefore implemented the strict SOP form from the prime implicants chart for the comparator in the given archtecture. Note I ommitted several of the muxes from most of the CLBs as they were never used. Apologies for the handwritten labels.



$\overline{O_7} \cdot O_3 + \overline{O_7}\,\overline{O_6}\,O_2$
$G_1 + \overline{O_5} \cdot F_1$
$O_3 \cdot O_2 \cdot O_1$
P_1
I_1
$O_3 \cdot O_2$
$O_6 \cdot G_4 + F_4$
$\overline{O_7} \cdot \overline{O_5} \cdot O_2 \cdot O_1$
I_3
P_4
$\overline{O_4} \cdot O_0 \cdot I_1 + \overline{O_4} \cdot O_0 \cdot I_2$
$G_2 + O_1 \cdot F_2$
$\overline{O_7} \cdot \overline{O_6} \cdot \overline{O_5}$
P_2
I_2
$O_3 \cdot \overline{O_6} \cdot \overline{O_5}$
$G_5 \cdot F_5$
$O_1 + \overline{O_4} \cdot O_0$
P_5
$\overline{O_7} \cdot \overline{O_4} \cdot O_0$
$G_3 \cdot F_3$
$O_2 \cdot O_1 + \overline{O_5} \cdot O_2 + \overline{O_6} \cdot O_1$
P_3
$\overline{O_5} \cdot \overline{O_4} \cdot I_3 \cdot O_0$
$G_6 + F_6 + P_5$
$P_1 + P_2 + P_3 + P_4$
LT
CLK
$\overline{LT} \cdot X$
$G_7 + F_7$
$LT \cdot a$
D
Q
$\overline{Q}$
X
00
01
10
11

8

```
################################################################################
# qm.py                                                                        #
#                                                                              #
# This file contains a Quine-McCluskey algorithm implementation.               #
# Quine-McCluskey outputs a simplified boolean expression given a truth table. #
#                                                                              #
# Author: Edward Speer                                                         #
#                                                                              #
# Revision History:                                                            #
# 10/9/2024  - Initial revision                                                #
# 10/14/2024 - Debug repeated implicants in recursion                          #
# 10/15/2024 - Add nice printing of prime implicant chart                      #
# 10/19/2024 - Print column numbers in PI tables                               #
################################################################################

################################################################################
# IMPORTS                                                                      #
################################################################################

from typing import List, Dict
from itertools import combinations, product
from re import findall

################################################################################
# HELPER FUNCTIONS                                                             #
################################################################################

"""
check_merge(minterm1: str, minterm2: str) -> bool

Check if two minterms can be merged. Minterms may be merged if they differ by a
single bit.
"""
def check_merge(minterm1: str, minterm2: str) -> bool:
    # Check that all dashes in minterms are in the same position
    if not [i for i, c in enumerate(minterm1) if c == '-'] == \
            [i for i, c in enumerate(minterm2) if c == '-']:
        return False

    # Obtain int representations of the minterms from binary strings
    minterm1_int, minterm2_int = list(map(lambda x:int(x.replace('-',
                                            '0'), 2), [minterm1, minterm2]))

    # To merge, only one bit can be twiddled
    xor = minterm1_int ^ minterm2_int
    return xor != 0 and xor & (xor - 1) == 0

"""
merge(minterm1: str, minterm2: str) -> str

Merge two minterms. The minterms must be able to be merged. The function returns
a new minterm with a dash in the position of the single twiddled bit from the
input minterms.
"""
def merge(minterm1: str, minterm2: str) -> str:
    return "".join(list(map(lambda x, y: x if x == y else '-', minterm1,
                            minterm2)))

################################################################################
# Logic Problem Class                                                          #
################################################################################

"""
LogicProblem

This class represents a logic problem. It contains a set of minterms for the
problem, and holds methods for finding prime implicants and creating a prime
implicant chart. This will allow for the creation of a simplified boolean
expression.
"""
class LogicProblem():

    # CONSTRUCTORS

    """
    __init__(self, active: List[str], inactive: List[str], mutex: bool=False)
```

```
Given a list of active and inactive minterms, create a new LogicProblem
object. The active minterms are the minterms that are true in the truth
table, and the inactive minterms are the minterms that are false in the
truth table. The false minterms are used to collect the don't care terms
for the problem. If mutex (mutally exclusive) is set to True, the inactive
set will automatically be set to the complement of the active set.
"""
def __init__(self, active: List[str], inactive: List[str],
             mutex: bool=False):
    self.minterms = active
    self.vars = len(active[0])

    if mutex:
        self.super = self.minterms
    else:
        self.super = list(set([''.join(bits) for bits in product('01',
                      repeat=self.vars)]) - set(inactive))

# METHODS

"""
prime_implicants() -> None

This function takes a list of minterms and returns a list of fully merged
prime implicants. The implementation is recursive, calling itself until a
traversal of the minterms has been completed without any further merging.
"""
def get_prime_implicants(self, minterms=None) -> None:

    # If this is the first iteration, start from all active + don't cares
    if minterms is None:
        minterms = self.super

    # list of prime implicants
    p_is = []

    # List of minterms that have been merged (one bool per minterm)
    merged = {minterm: False for minterm in minterms}

    # Merge any possible minterms from all combinations and count merges
    merge_cnt = list(map(
        lambda t: (p_is.append(merge(t[0], t[1])) or
                ((lambda i, j: merged.update({i: True}) or
                    merged.update({j: True}) or False)(t[0], t[1])) or
                True) if check_merge(t[0], t[1]) and merge(t[0], t[1]) not
                in minterms else False,
                list(combinations(minterms, 2)))).count(True)

    # Add any unmerged terms to the prime implicants list
    p_is += list(set([minterm for minterm in minterms if not
                      merged[minterm]]))

    # If no merges were made, return the prime implicants list. Otherwise,
    # call the function recursively.
    self.p_is = list(set(p_is))
    if merge_cnt != 0:
        self.get_prime_implicants(minterms=self.p_is)

"""
p_i_chart() -> None

Create a prime implicant chart. The chart is a dictionary with the prime
implicants as keys and the minterms that the prime implicant covers as
values.
"""
def p_i_chart(self) -> None:
    self.p_i_dict = {p_i.replace("-", "."): "" for p_i in self.p_is}
    for minterm in self.minterms:
        for p_i in self.p_i_dict:
            if findall(p_i, minterm):
                self.p_i_dict[p_i] += "1"
            else:
                self.p_i_dict[p_i] += "0"
    self.p_i_dict = {k: self.p_i_dict[k] for k in sorted(self.p_i_dict) if
                    "1" in self.p_i_dict[k]}

"""
```

```
run_qm() -> None

Run the Quine-McCluskey algorithm on the logic problem. This function will
call the get_prime_implicants() and p_i_chart() functions, and will print
the prime implicant chart.
"""
def run_qm(self)->None:
    self.get_prime_implicants()
    self.p_i_chart()

"""
get_table() -> None

Print the prime implicant chart in a table format.
"""
def get_table(self)->None:
    table_len = (5 * len(self.minterms) + self.vars + 4)
    print("=" * table_len)
    print(f"| {' ' * len(self.minterms[0])} ", end="")
    for i in range(1, len(self.minterms) + 1):
        if (i < 10):
            print(f"| {i} |", end="")
        else:
            print(f"|{i} |", end="")
    print("")
    print("-" * table_len)
    for p_i in self.p_i_dict:
        coverage = ["| x |" if c == "1" else "|   |" for c in
                    self.p_i_dict[p_i]]
        print(f"| {p_i} " + "".join(coverage))
        print("-" * table_len)
    print("=" * table_len)
```

```
################################################################################
# This file uses the Quine-McCluskey algorithm implemented in the Q-M module   #
# to solve for the prime implicants chart for each segment in the comparator   #
# for probelm 13 and 14 of HW5 in EECS119a.                                    #
#                                                                              #
# Author: Edward Speer                                                         #
# Revision History:                                                            #
# 11/5/2024 - Initial Revision                                                 #
################################################################################

################################################################################
# CONSTANTS                                                                    #
################################################################################

# Path to Quine-McCluskey algorithm implementation
QM_PATH = "../Q-M"

# This dictionary will hold the active and inactive inputs for the comparator
INPUTS = {
    "comparator": {
        "active":   [],
        "inactive": []},
}

################################################################################
# IMPORTS                                                                      #
################################################################################

import sys

# Add the path to the Quine-McCluskey algorithm to the system path
sys.path.append(QM_PATH)
import qm

################################################################################
# MAIN                                                                         #
################################################################################

"""
main()

The main function for this script runs the Quine-McCluskey on each of the
defined input tables in the INPUTS dict. The resulting prime implicants chart is
printed on stdout.
"""
def main():
    # Generate all 8 bit input values possible
    for i in range(256):  # 2^8 = 256 possible 8-bit combinations
        # Convert the number to an 8-bit binary string
        binary_string = f"{i:08b}"

        # Split the string into high nibble and low nibble
        high_nibble = int(binary_string[:4], 2)
        low_nibble = int(binary_string[4:], 2)

        # Check if high nibble is less than low nibble
        if high_nibble < low_nibble:
            INPUTS["comparator"]["active"].append(binary_string)
        else:
            INPUTS["comparator"]["inactive"].append(binary_string)

    for segment in INPUTS:
        print(f"Signal: {segment}")
        active = INPUTS[segment]["active"]
        inactive = INPUTS[segment]["inactive"]
        lp = qm.LogicProblem(active, inactive)
        lp.run_qm()
        lp.get_table()

if __name__ == "__main__":
    main()
```

Prime implicants of signal: Comparator

```
======================================================================================================================================================================
|          | 1 || 2 || 3 || 4 || 5 || 6 || 7 || 8 || 9 ||10 ||11 ||12 ||13 ||14 ||15 ||16 ||17 ||18 ||19 ||20 ||21 ||22 ||23 ||24 ||25 ||26 ||27 ||28 ||29 ||30 ||31 |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| ...01111 |   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| ..0.111. |   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| ..0011.1 |   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| .0..11.. |   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x || x || x ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| .0.01.11 |   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| .00.1.1. |   ||   ||   ||   ||   ||   ||   ||   ||   || x || x ||   ||   || x || x ||   ||   ||   ||   ||   ||   ||   ||   || x || x ||   ||   || x || x ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| .0001..1 |   ||   ||   ||   ||   ||   ||   ||   || x ||   || x ||   || x ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 0...1... |   ||   ||   ||   ||   ||   ||   || x || x || x || x || x || x || x || x ||   ||   ||   ||   ||   ||   || x || x || x || x || x || x || x || x ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 0..0.111 |   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 0.0..11. |   ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   || x || x ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   || x || x ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 0.00.1.1 |   ||   ||   ||   || x ||   || x ||   ||   ||   ||   ||   || x ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 00...1.. |   ||   ||   || x || x || x || x ||   ||   ||   ||   || x || x || x || x ||   ||   || x || x || x || x ||   ||   ||   ||   || x || x || x || x ||   || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 00.0..11 |   ||   || x ||   ||   ||   || x ||   ||   ||   || x ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 000...1. |   || x || x ||   ||   || x || x ||   ||   || x || x ||   ||   || x || x || x || x ||   ||   || x || x ||   ||   || x || x ||   ||   || x || x ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| 0000...1 | x ||   || x ||   || x ||   || x ||   || x ||   || x ||   || x ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
======================================================================================================================================================================
================================================================================================================================================================
|32 ||33 ||34 ||35 ||36 ||37 ||38 ||39 ||40 ||41 ||42 ||43 ||44 ||45 ||46 ||47 ||48 ||49 ||50 ||51 ||52 ||53 ||54 ||55 ||56 ||57 ||58 ||59 ||60 ||61 ||62 ||63 |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   || x || x || x || x ||   ||   ||   ||   ||   ||   ||   ||   || x || x || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   || x ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   || x || x || x || x || x || x || x || x ||   ||   ||   ||   || x || x || x || x || x || x || x || x ||   ||   ||   || x || x || x || x || x || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   || x ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   || x ||   ||   ||   ||   ||   || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
| x || x || x ||   ||   ||   ||   || x || x || x || x || x || x || x || x ||   ||   ||   ||   || x || x || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   || x ||   ||   ||   || x ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
================================================================================================================================================================
================================================================================================================================================================
|64 ||65 ||66 ||67 ||68 ||69 ||70 ||71 ||72 ||73 ||74 ||75 ||76 ||77 ||78 ||79 ||80 ||81 ||82 ||83 ||84 ||85 ||86 ||87 ||88 ||89 ||90 ||91 ||92 ||93 ||94 ||95 |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
| x || x ||   ||   ||   ||   ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   || x |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
| x || x ||   ||   || x || x || x || x || x || x || x || x ||   || x || x || x || x || x || x || x || x || x || x || x || x || x || x || x || x ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
| x || x || x || x ||   ||   ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
----------------------------------------------------------------------------------------------------------------------------------------------------------------
================================================================================================================================================================
=============================================================================================================================
|96 ||97 ||98 ||99 ||100 ||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120|
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   ||   ||   ||   ||   ||   || x ||   ||   || x |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   || x || x ||   ||   ||   ||   || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x || x || x || x ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   || x ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   || x ||   || x ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
| x || x || x || x ||   ||   || x || x || x || x ||   || x || x || x || x || x || x || x || x ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   || x ||   ||   ||   ||   ||   ||   || x ||   ||   ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   || x || x || x || x ||   ||   || x || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   || x ||   || x ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
|   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   |
-----------------------------------------------------------------------------------------------------------------------------
=============================================================================================================================
```