Team 14 | Mai 18, 2017

Logic Design Project

Ic Tester

# Submitted to:

Eng/ Youssef Ghatas

Eng/ Marwa

Eng/ Dina

# Submitted by:

1. Abdelrahman Mohamed
2. Aisha Mousa
3. Hady Maher
4. Ahmed Adel

# Introduction

This project test 4 ICs (AND, OR, XOR, NAND) with 2 bit input, We change the main design in project description file.

Our new circuit is designed to escape with the first wrong output, we use 2 ICs with 2 bit counter, 1 \* 16 MUX, 1 IC AND, 1 IC OR, 1 IC NOT, 1 IC XOR.

# Input Map

|  |  |
| --- | --- |
| Input 1 | Reset the circuit |

# Output LEDs Map

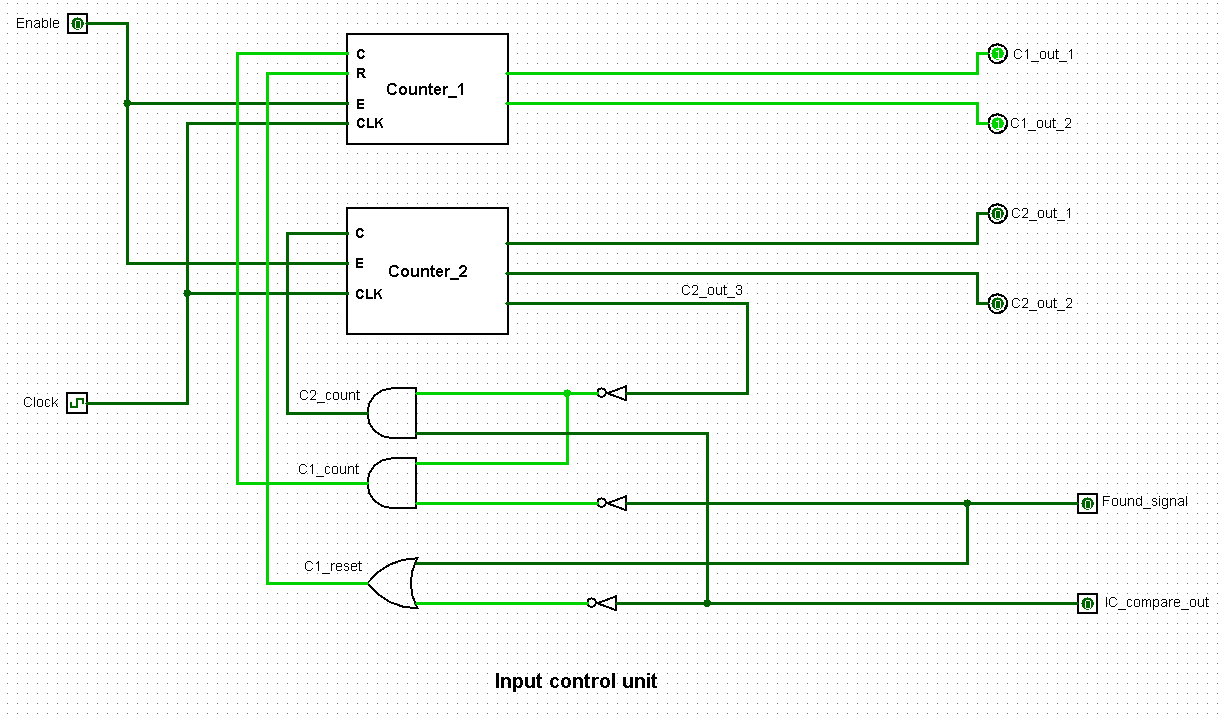
|  |  |
| --- | --- |
| LED 1 | Found flag |
| LED 2 | Not found flag |
| LED 3 & 4 | Number of IC |
| LED 5 | Truth table output |
| LED 6 & 7 & 8 & 9 | Output of tested IC |

# ICs

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Number | Expansion |
| 74161 | Counter | 2 | 2 |
| 74150 | 16 \* 1 MUX | 1 | - |
| 7408 | AND | 1 | - |
| 7404 | NOT | 1 | 3 |
| 7486 | XOR | 1 | - |
| 7432 | OR | 1 | - |

## IC Tester

## INPUT CONTROL UNIT

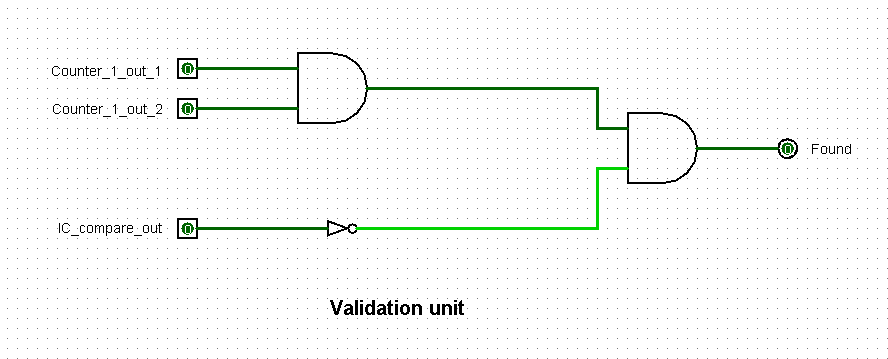
****

## Truth Table unit

## 

## IC Compare

## Validation unit

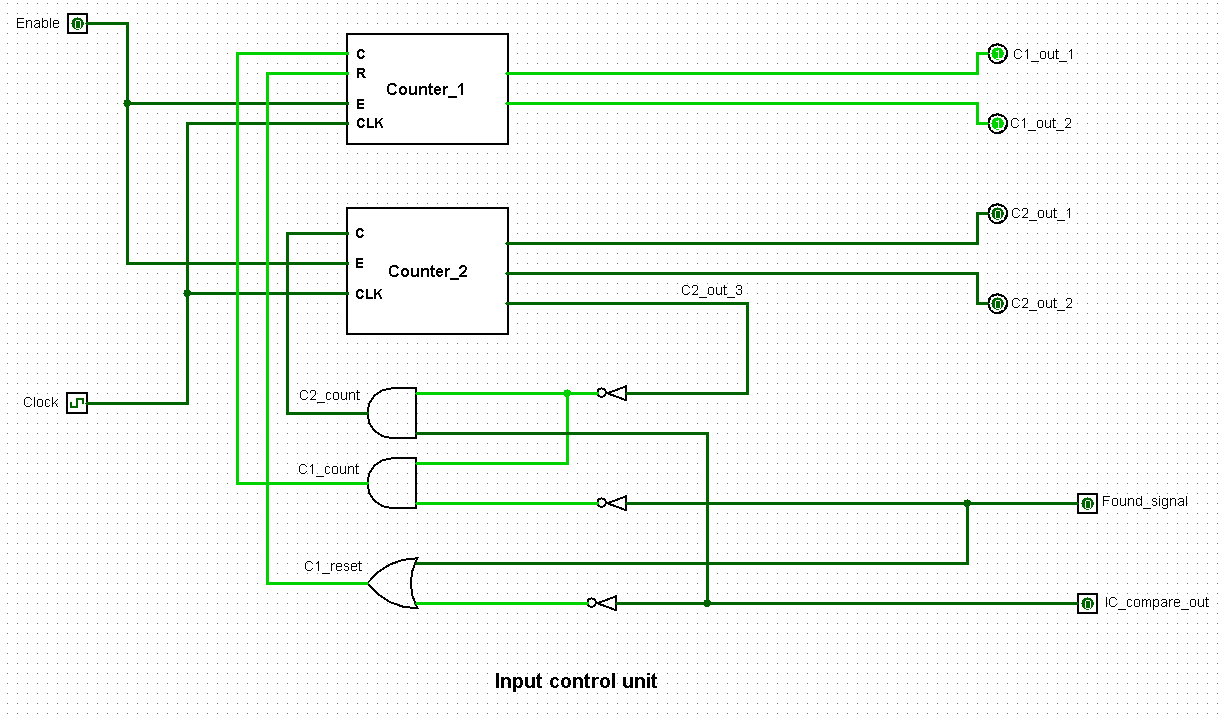
****

# Project description

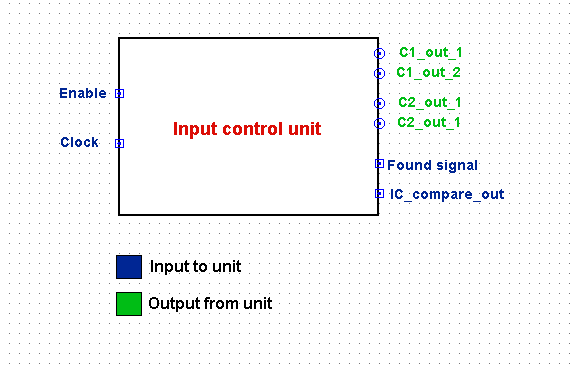
In this project, we are making an IC tester for 2 terminals ICs. It’s required to compare all IC gates outputs with search table to indicate if the output matches any of the valid ICs. If the IC was found the entire circuit pause and Found signal is triggered. If the IC was not in the entire search table then the entire circuit pause and NotFound signal is triggered. So basically, we have 4 units: 1- Input control unit 2- Search table unit 3- IC\_compare unit 4- Validation unit.

## 1. Input control unit

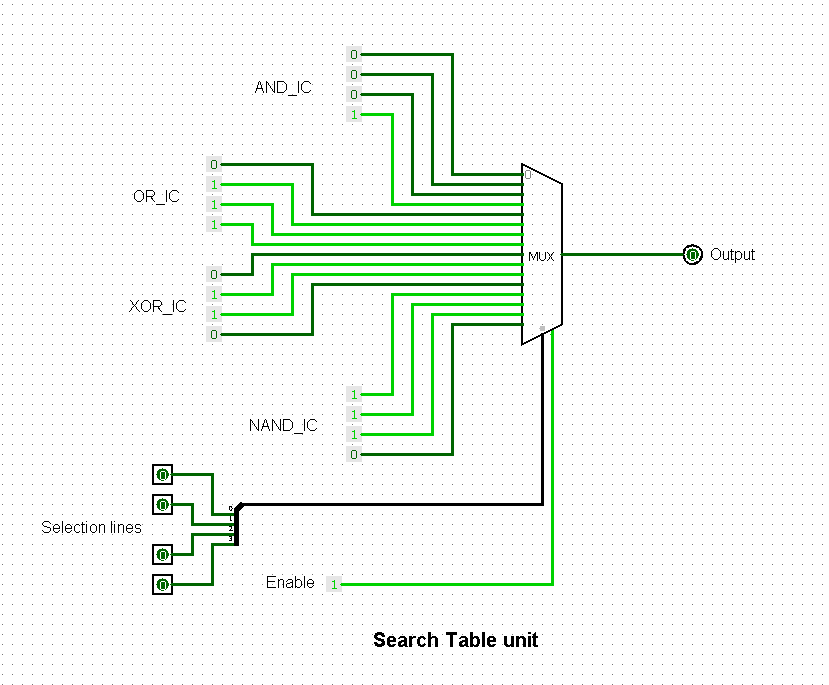
This unit is supposed to handle all inputs in the circuit. So, it has only one input which Enable to enable and disable the entire circuit. It consists of two 2-bit counters and a small combinational circuit. Counter\_1 which is responsible for the input of the IC. Counter\_2 only counts when Counter\_1 reaches 4 or when the output of the IC\_compare equals when which means that the output of the IC didn’t match for this case so, we skip the entire case (by resting counter\_1 to zero). When Counter\_2 reaches 4 and the IC is not found that mean that the IC is not valid so, the entire circuit pause and NotFound signal is triggered. Both Counter\_1 and Counter\_2 outputs are used as index for the search table. The unit takes 2 other inputs from the circuit as a feedback which are Found signal and IC\_compare output signal. Found signal pass throw NOT gate then AND with Counter\_2 third output and the result is used to enable Counter\_1 to count which means that if the IC is not found and Counter\_2 didn’t finish counting then Counter\_1 should continue counting. IC\_compare output signal (Note: IC\_compare output zero means true, one means false) is used two times. First it passes throw NOT gate and OR with Found signal which means that if the output of the gates still matches the case or the IC was found then Counter\_1 (Note: Reset equal zero the counter reset to zeros Reset equal one the counter does not Reset) is not reset to zero. Finally, the third output of Counter\_2 passes throw AND with the IC\_compare signal after it pass throw NOT gate and the result is used to enable counting of Counter\_2 which means that Counter\_2 counts when it didn’t reach 4 yet and IC\_compare was false. So the final design would look like the following



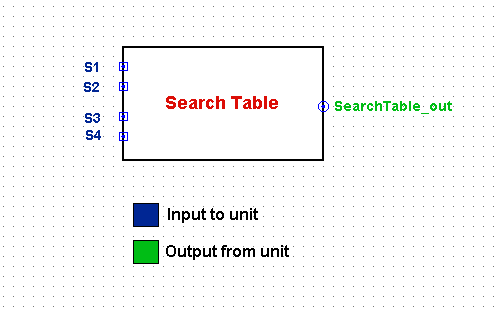
Taking the unit as a sub circuit with inputs and outputs to make it easier to understand it would look as the following



## 2. Search Table unit

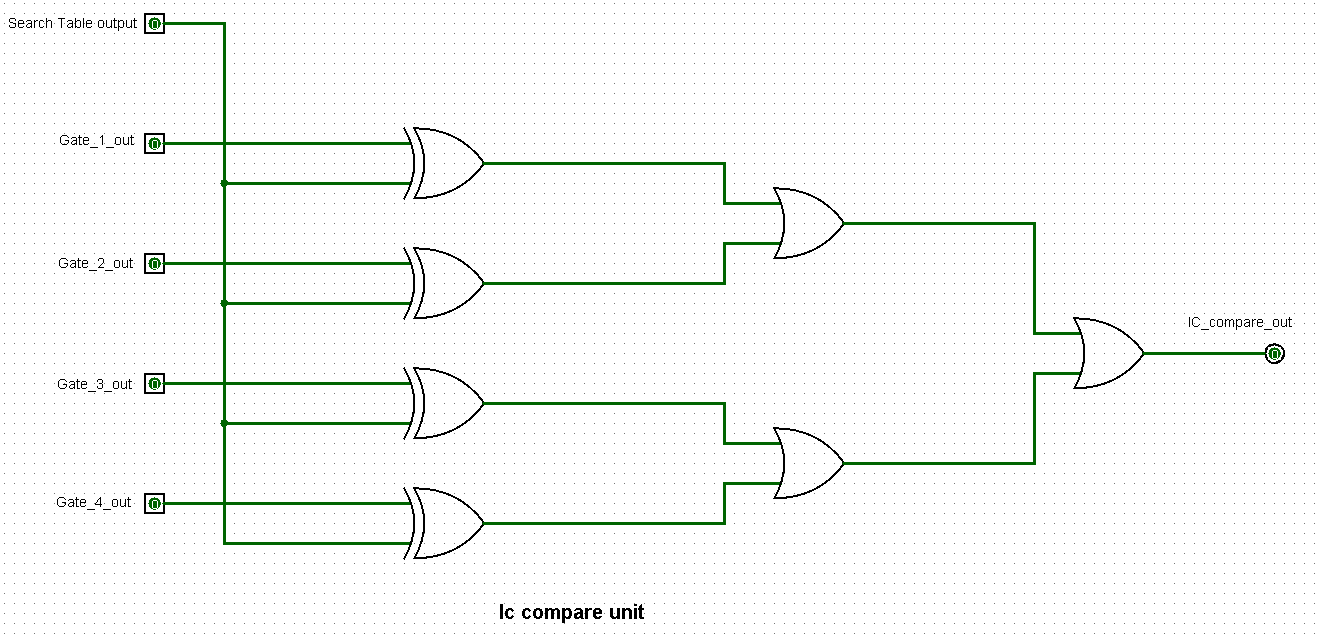
This unit is responsible for giving output of the valid IC in sequence to give the output to IC\_compare unit to compare it with the output of the IC. From the previous specification then the unit would take a fixed input and give one output using index. This functionality matches a Multiplexer and scene we have only 4 valid ICs it would be 16x1 Multiplexer as every IC has a truth table of 4 row so the entire table would be 16x1. From previous description, the unit would be the following

Treating it a discrete unit with inputs and outputs it would look as the following

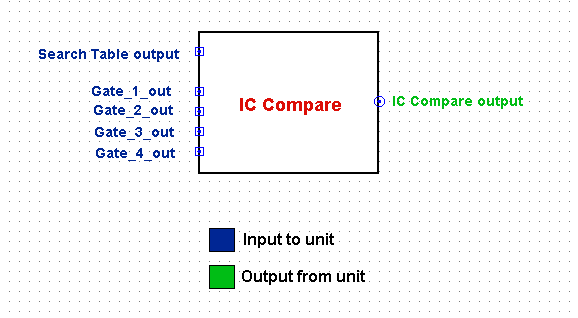


## 3. Ic compare unit

This unit is supposed to compare between IC gates outputs and Search Table output to detect if the output of the gates matches any of the 4 valid ICs. So, we need to check equality which is the function of XOR. The design then would be 4 gates XOR each one takes output of each IC gate and Search Table output would be the second input of the XOR. So, any XOR gives 1 means that the comparison yield to false and the current case is not true. In conclusion after XORs we need to OR them all so that if any XOR gives 1 then the entire comparison is false. From this specification, the design would be the following.

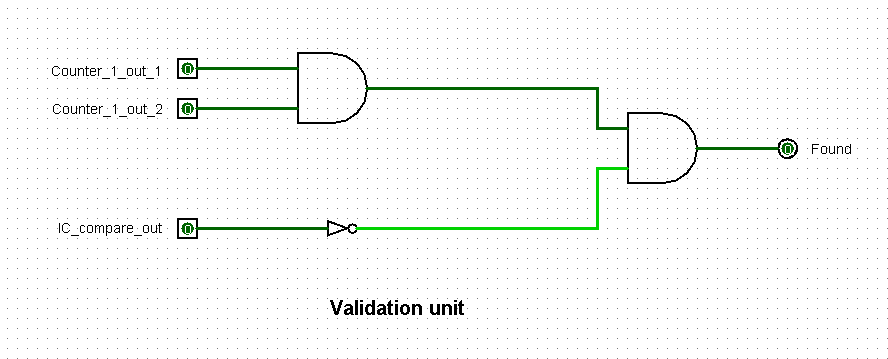


The unit as block with inputs and outputs would be as the following

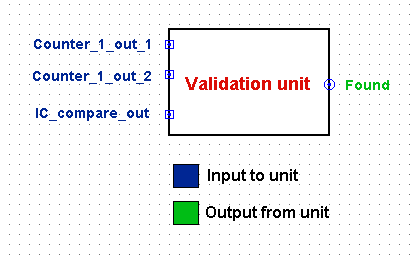


## 4. validation unit

This is the last unit and it’s used to make a simple check if the IC was found or not. So, it takes Counter\_1 outputs and IC\_compare output as inputs. If Counter\_1 output is 3 and IC\_compare is true then that means that the IC gave 4 outputs exactly as the current case then the IC is valid and it’s the current case. It very trivial we simply would pass IC\_compare output throw NOT then AND it with both Counter\_1 outputs the result would the Found signal. From this specification, the design would be the following



The unit as a block with inputs and outputs would be the following



At the end the full design would be the following

