# 

# **Experiment 2**

**Combinational Logic Design with**

# **NAND & NOR Implementations**

**Objective**

* To practice with designing and implementing different functions

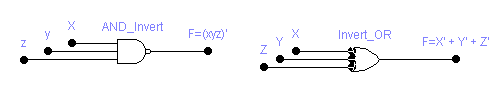
with only NAND gates or NOR gates.

* Therefore we can conclude the NAND & NOR gates are universal.

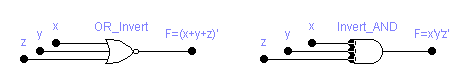
**Basic concepts**

Digital circuits are more frequently constructed with NAND or NOR gates than with AND/OR gates because these gates are easier to fabricate with electronic components and they are the basic gates over all digital logic IC families. In consequence of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from Boolean functions in terms of AND, OR and NOT into equivalent NAND or NOR logic diagrams.

To facilitate the conversion to NAND and NOR logic, it is convenient to know two graphic symbols for these gates (see Figure 1) that are different from the standard ones.



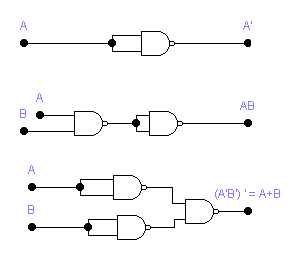
a: Two graphic symbols for NAND gate



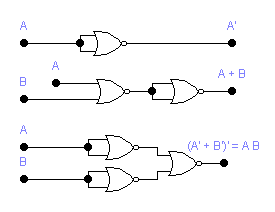
b: Two graphic symbols for NOR gate

Figure 1

NAND and NOR gates are said to be universal gates because any digital system can be implemented with one type of these gates. Figure 2 shows the implementation of NOT, AND & OR gates using only NAND or NOR gates.



a: Implementation of NOT, AND, and OR **using NAND** gates



b: Implementation of NOT, OR, & AND **using NOR** gates

Figure 2: NAND and NOR gates are universal gates

The techniques for implementing Boolean functions or circuits with NAND or NOR gates depend on the number of levels of the network.

**The procedure for designing a minimum NAND network that implements a function F:**

Assuming that all variables and their complements are available as inputs:

1. Find a minimum sum-of-product expression for F.

2. Draw the corresponding AND-OR-Invert network.

3. Replace all gates with NAND gates leaving the gate interconnections unchanged. If the output gate has any single literals as inputs, complement these literals.

**The following method can be used to realize F with NOR gates:**

Assuming that all variables and their complements are available as inputs:

1. Find a minimum product-of-sums expression for F.

2. Draw the corresponding two-level OR-AND-Invert network.

3. Replace all gates with NOR gates leaving the interconnections unchanged.

If the output gate has any single literals as inputs, complement these

literals.

**Procedure**

1. Design the following using **Workbench only**:

Suppose we want to compare two unsigned two-bit integers together:

*a*[1:0] and *b*[1:0].

Only P0 = 1 if A > B.

Only P1 = 1 if A = B.

Only P2 = 1 if A < B.

Complete the following truth table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1st Number** | | **2nd Number** | | **Product** | | |
| *a1* | *a0* | *b1* | *b0* | *P2* | *P1* | *P0* |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | o | 0 |
| 0 | 0 | 1 | 0 | 1 | o | 0 |
| 0 | 0 | 1 | 1 | 1 | o | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |  |  |  |
| 0 | 1 | 1 | 0 |  |  |  |
| 0 | 1 | 1 | 1 |  |  |  |
| 1 | 0 | 0 | 0 |  |  |  |
| 1 | 0 | 0 | 1 |  |  |  |
| 1 | 0 | 1 | 0 |  |  |  |
| 1 | 0 | 1 | 1 |  |  |  |
| 1 | 1 | 0 | 0 |  |  |  |
| 1 | 1 | 0 | 1 |  |  |  |
| 1 | 1 | 1 | 0 |  |  |  |
| 1 | 1 | 1 | 1 |  |  |  |

2. Optimize P1&P2 K-maps, draw the And-Or-Invert designs then convert

them to an only NOR gates circuit and simulate each function separately

**by gates** using EWB.

1. Optimize the equation for P0 using a K-map, draw the And-Or-Invert

design then convert it to an only NAND gates circuit and simulate it **by gates** using EWB.

**Note: You have to design your three outputs in AND OR INVERT 1st**

**then transfer your designs into NAND only or NOR only.**

**The circuits in steps 2 and 3 should include a numeric display**

**for each of the two numbers you compare and a LED for**

**each of P0, P1 and P2 outputs.**

B. Design using EWB **in gates for the output sum and in I.C.’s for the output**

**carry** and implement your design on the lab board ( for output carry only )

for the full adder using **2 inputs NAND gates only.**

C. Design using EWB in **in gates for the output sum and in I.C.’s for the output**

**carry** and implement your design on the lab board ( for output carry only )

for the full adder using **2 inputs NOR gates only.**

D. Design using EWB in I.C.’s and implement your design on the lab board for the

logic circuit that generates an odd parity bit output for its three bits input

message using **2 input NAND gates only.**

E. Design using EWB in I.C.’s and implement your design on the lab board for the

logic circuit that generates an odd parity bit output for its three bits input

message using **2 input NOR gates only.**

F. Design using EWB in I.C.’s and implement your design on the lab board for the

logic circuit that generates an output “1” if its three inputs have only

one input equals “1” and “0” otherwise using **2 inputs NAND gates only**.

G. Design using EWB in I.C.’s and implement your design on the lab board for the

logic circuit that generates an output “1” if its three inputs have only

one input equals “1” and “0” otherwise using **2 inputs NOR gates only**.