A Novel Two – Input NAND Gate Design for Implementing the AND, OR & XOR Gates for Digital Circuits

Ofoegbu .O. Edward

Department of Electrical Electronics Engineering, Adeleke University, Ede, Osun-State, Nigeria

***Abstract*:-The research paper discusses a novel two input NAND gate design for implementing standard logic functions used in digital logic circuits. Various researchers have proposed designs and implementation of logic functions using NAND and NOR gate circuits but this paper proposes a three-NAND gate architecture for implementing AND & OR logic as well as a four-NAND gate architecture for implementing the XOR logic. A minimization of the number of NAND gates required to implement a logic function was achieved though there are other designs, this architecture provides for smooth logic transition.**

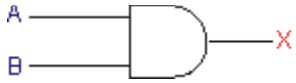
***Keywords*: NAND, NOR, OR, AND, Digital Circuits**

I. INTRODUCTION

Digital Circuit is entirely constructed from an infinitesimal number of different types of building block circuits. These circuits have their respective one or more input connections and one output connection. Signals of special characteristics are applied to the inputs to bring about an output, which is observed by the help of a test meter. The signals are always at one or the other of two voltage levels. Succinctly, these voltage levels can be referred to as high or low voltage. The signals are called **logic signals** and the circuits, **logic circuits**. Also, they are called **true or false** denoted by the state **'1' or '0'** respectively and the circuits as **logic circuit**. Therefore, Digital Circuits are the representation of functions that convey information about the behaviour or attribute of an electronic by discrete bands of analog levels, rather than by a continuous range. **Logic gates** of simple electronic representations of Boolean logic functions in their large arrays or assemblies are used to develop digital electronic circuits.

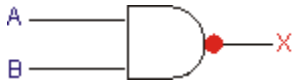
Logic Gates: A logic gate is an apparatus that uses a **logical operation** to perform an output. Logic gates in another word are defined as simple electronic representations of Boolean logic functions. We have about seven (7) gates with different operations and applications, namely: AND, OR, XOR, NAND, NOR, XNOR, and NOT gates. Below are the diagrams of the logic gates with their truth tables and explanations:

AND gate: The AND gate is a **logic gate** that implements **logical conjunction**. According to its behaviour, A HIGH output '1' is produced if only both the inputs to the AND gate are HIGH and if neither or only one input to the AND gate is HIGH, a LOW output is produced. With this, more of the outputs are '0' except when the inputs are '1s'. Therefore, we say AND gate finds the *minimum* i.e. between two binary digits. (Output is always '0').



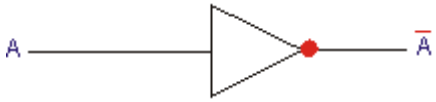
INPUT		OUTPUT
A	**B**	**A AND B**
0	0	0
0	1	0
1	0	0
1	1	1

NAND gate: A NAND gate which simply means **Negated AND or NOT AND** is a logic gate which produces an output that is false only if its inputs are true; therefore, we say its outputs is **complement** to, **or opposite** of that of the **AND gate**. If both inputs are HIGH (1), a LOW (0) output results while a HIGH output results when one or both inputs are LOW. The NAND gate has the property called **functional completeness** because of its significance which proves that any Boolean function can be implemented by using a combination of NAND gates.



INPUT		OUTPUT
A	**B**	**A NAND B**
0	0	1
0	1	1
1	0	1
1	1	0

NOT gate (An inverter): Simply, the **NOT gate** is a logic gate which implements **logical negation**.



INPUT	OUTPUT
A	NOT A
0	1
1	0

OR gate: The OR gate is a **logic gate** that implements **logical disjunction**. According to its behaviour, A HIGH output '1' is produced if one or both the inputs to it are HIGH and if neither input is HIGH, a LOW output is produced. With this, more of the outputs are '1' except when the inputs are '0s'. Therefore, we say OR gate finds the *maximum* i.e. between two binary digits. (Output is always '1').

INPUT		OUTPUT
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

NOR gate: It is a digital logic gate that implement logical NOR. It behaviour explains a HIGH output being produced when both inputs to it are LOW; a LOW output results, if one or both input is HIGH. The **negation of OR operator** gives us the NOR. Looking at its truth table below, the NOR gate can be seen as an AND having all its input inverted. It also has the **functional completeness** property just like that of NAND gate.

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

XOR gate: The XOR gate which is sometimes pronounced as EOR or EXOR (**i.e. Exclusive OR gate**) is a digital logic gate that performs an exclusive or; i.e. HIGH output is produced if one and only one of the inputs to the gate is HIGH or TRUE. A FALSE output results, if both inputs are FALSE and TRUE. The XOR gate expresses the **inequality function** i.e. if the inputs are not alike; it produces an output that is TRUE.

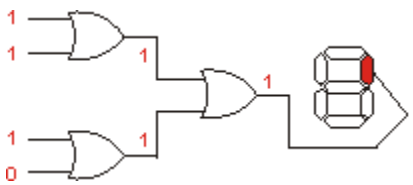
INPUT		OUTPUT
A	B	A XNOR B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR gate: The XNOR gate is simply the **reverse** of an XOR gate in which the output is the negation of the normal output of an XOR gate. In the XNOR gate, the output can only be TRUE, if the inputs are alike ('1' XNOR '1'/'0' XNOR '0'). The reverse is the case to make the output FALSE i.e. the inputs are not alike.

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	0
1	0	0
1	1	1

Application Areas of These Gates:

So far, we have discussed the various digital logic gates: the design, property and operation of the gates and also, we will be looking at their applications. But precisely, we will be examining OR gate applications. Below is the diagram showing the application of an OR gate:

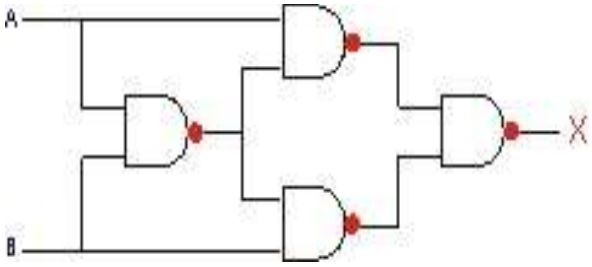


This diagram showing the application of an OR gate is well explanatory. We can see that one of the seven segments display is powered because of the output state of '1 value' which simply means ON. But when the output is of the state '0 value', there will be no display, which means OFF. The application above is determined mainly based on the OR gate truth table or mode of operation and other logic gates applications are based on their truth table either. NB: In the above diagram. Other LEDs can be powered depending on where the output is conveyed to.

II. REVIEW OF LITERATURE

Nathan (2013) et al, developed an XOR gate using a four (4) two input NAND gates. The two inputs go to the first NAND gate to produce an output which serves as both the second

input for the middle – top NAND gate and first input for the middle – bottom NAND gate and the previous inputs A and B make up for the first and second inputs of the middle – top and bottom NAND gate respectively.



Suresh (2007) proved how he came about the XOR implementation using AND gate and NOR gate, and inputs were given to the two gates and there outputs were connected to another NOR gate. So altogether 6 NAND gates were used for the XOR implementation.

Blake (2002), proved how he drove out XOR gate using NAND gate that assuming that there are two inputs and no other constant inputs, you can create an XOR gate using NAND gates by:

C = (A NAND B)D = ((C NAND C) NAND C)

XOR = ((A NAND D) NAND B) NAND ((B NAND D) NAND D) NAND A)

Now, Testing the NAND gates

A = 0, B = 0, C = (A NAND B) = (0 NAND 0) = 1

D = (C NAND C) NAND C = (1 NAND 1) NAND 1 = 0 NAND 1 = 1.

Barrie (1998) et al, explained that Ex – OR function is not a basic logic gate but a combination of different logic gates connected together. He said, using the 2 – input truth table of the XOR, we can expand the XOR function to: (A+B).(A.B') which means that, we can realize this new expansion using the following individual gates below (fig. 1). He further explained that one of the main disadvantage of implementing the XOR function is that it contains three different types logic gates OR, NAND, and finally AND within its design. Then He said one of the easier ways of producing the XOR function from a single gate is to use our old favourite, which is the NAND gate (fig. 2)

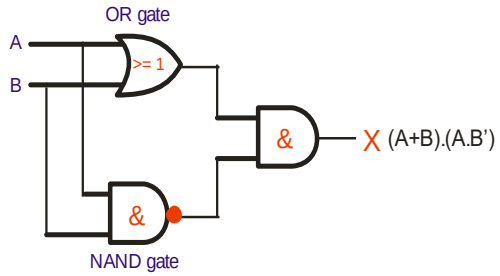


Fig. 1

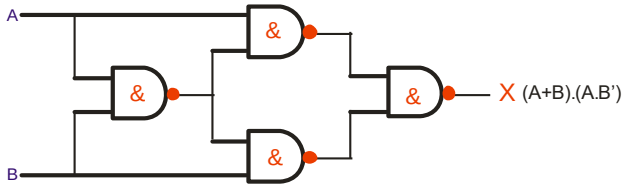
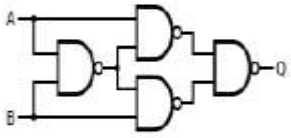
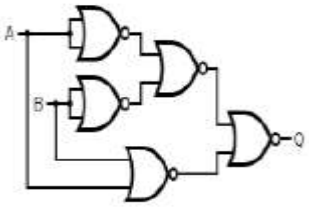


Fig. 2

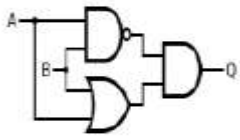
Fletcher, (1980) explained that, if a specific type of gate is not available, a circuit that implements the same function can be constructed from other available gates. A circuit implementing an XOR function can be trivially constructed from an XNOR gate followed by a NOT gate. If we consider the expression we can construct an XOR gate circuit directly using AND, OR and NOT gates. However, this approach requires five gates of three different kinds. He said, An XOR gate circuit can be made from four NAND or five NOR gates in the configurations shown below. In fact, both NAND and NOR gates are so-called "universal gates," and any logical function can be constructed from either.



XOR gate circuit constructed using only NAND gates.

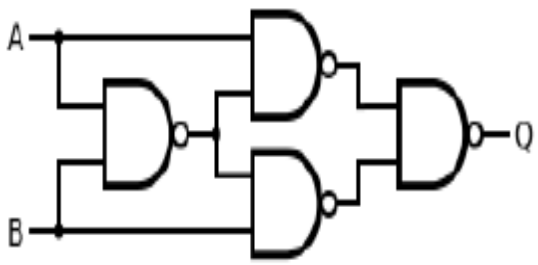


XOR gate circuit constructed using only NOR gates.

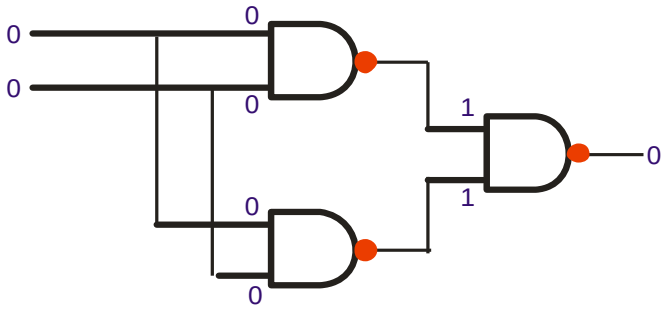


XOR gate circuit using three mixed gates

Donn (2010) developed an XOR gate using a four (4) two input NAND gates. The two inputs go to the first NAND gate to produce an output which serves as both the second input for the middle – top NAND gate and first input for the middle – bottom NAND gate and the previous inputs A and B make up for the first and second inputs of the middle – top and bottom NAND gate respectively.

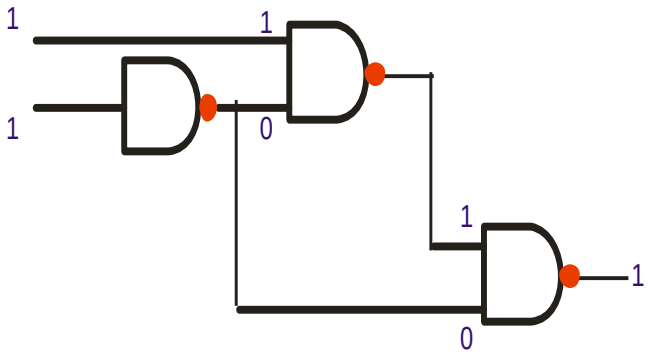


III. AND & OR GATE IMPLEMENTATION USING 3 NAND GATES



3 NAND gates working as an AND gate

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

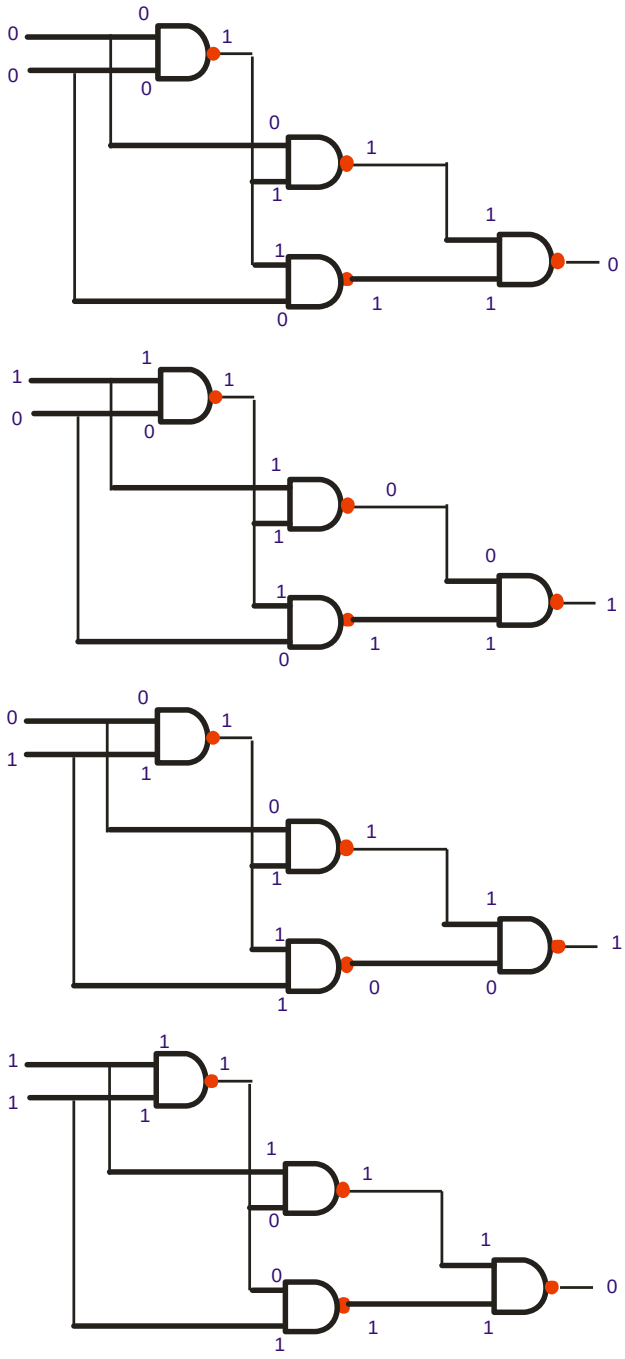


3 NAND gates working as an OR gate

INPUT		OUTPUT
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

The architectures shown above implements the truth tables for the AND & OR gates. Where in the first architecture the output for logic "low" applied at A and B respectively generated logic "high" after passing through the first NAND gate arrangement, when applied to a NAND gate will generate a logic "low" which is similar to what is obtainable in an AND gate. If the rest of the logic levels are applied it is observed that the truth table for an AND gate is realised. The same applies for the OR gate implementation.

IV. XOR GATE IMPLEMENTATION USING 4 NAND GATES



INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

The exclusive-OR gate logic was also realised using a 4-NAND gate arrangement as shown above. The sequence of occurrence of each logic level generated a final output for different inputs similar to the truth table for the XNOR.

V. CONCLUSION AND RECOMMENDATIONS

A new implementation of the AND, OR and XOR logic was developed in this paper. It utilized NAND gates which are universal gates to mimic the logic operations of the gates described earlier. A key consideration was to use fewer gates than other researchers to achieve the same results. The researchers hope to apply in future another universal gate (NOR) to the same architecture and see if the truth tables for the emulated gates are still generated.

REFERENCES

[1]. Anderson JC, Voigt CA, Arkin AP. Environmental signal integration by a modular AND gate. Mol. Syst. Biol. 2007;3:133. [PMC free article] [PubMed].

[2]. **Mano, M. Morris and Charles R. Kime**. *Logic and Computer Design Fundamentals, Third Edition.* Prentice Hall, **2004**. p. 73 .

[3]. Hyperphysics.phy-astr.gsu.edu. Retrieved 2012-09-24.

[4]. **Harris, David Harris, Sarah (2007)**. *Digital design and computer architecture* (1st ed. ed.). San Francisco, Calif.: Morgan Kaufmann. p. 21. 1SBN 9780123704979

[5]. **Brumbach, Michael E**. *Industrial electricity* (8th ed. ed.). Clifton Park, N.Y.: Delmar. p. 546. ISBN 9781435483743.

[6]. **Fletcher, William (1980)**. *An engineering aproach to digital design*. Prentice-Hall. p. 98. ISBN 0132776995

[7]. **P. K. Lala,** *Practical Digital Logic Design and Testing*, Prentice Hall, 1996,

[8]. **John Bird (2007)** *Engineering mathematics* Newness p.532. 1SBN 978-0-7506-8555-9.

[9]. **Hans kleine Buning: Theodor Lettmann (1999)**. Propositional logic: deduction and algorithms Cambridge University Press. p. 2. ISBN 978-0-521-63017-7.

[10]. **Jaeger**, *Microelectronic Circuit Design*, **McGraw-Hill** 1997, ISBN 0-07-032482-4 .pp. 226-233.

[11]. **Tinder, Richard F. (2000)**. *Engineering digital design: Revised Second Edition.* pp. 317-319. ISBN 0-12-691295-5 .Retrieved 2008-07-04.