MAT JOURNALS

Journal of Analog and Digital Devices
Volume 2 Issue 3

Logic Theory in Designing of Digital Circuit & Microprocessor

Prof.Vikram Mahendra Kakade
Assistant Professor, Electronics & Telecommunication Engineering Department,
Prof Ram Meghe College of Engineering & Management, Badnera-Amravati, Maharashtra India-444702
Email: vikramkakade.160@gmail.com

Abstract

This Paper Provides basic fundamentals to be used while designing logic gates in a digital circuit as we know that logic gates are nothing but the technique used for calculation of unknown's also provide how microprocessor instruction set is get affected due designing of logical gates. Instruction sets are used for simulative approach of digital circuit in Microprocessor which is later used in assembly language programming. Logic gates are constructed using diodes or transistors, but can also be constructed using electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or some of mechanical elements also plays important role in the construction of logic gates . Connecting output of fist logic gate into input of second create cascading with amplification, logic gates can be cascaded in the same way that Boolean functions can be composed, allowing the construction of a physical model of all of Boolean logic also paper deals with correlation between digital electronics and microprocessor

Keywords: *ANA R, Bits, Digitization, logic gates, TTL, ORA, XRA.*

INTRODUCTION

Process of converting continuous analog signals into a number of discrete states, a process is called as digitization, after which quantization is done to get digital signal as an output. Elimination of noise in digital signal is done using filter which has two basic principles

1. Storage over arbitrary periods of time flawless retrieval and reproduction of the stored information
2. Flawless transmission of the information

As we know digital signal are very easy to understand and most informative compared to analog signal also manipulating digital signal is very easy. But there are different types of error in digital signal like quantization error which creates difficulties in reproduction.

The development in digital techniques and technology has been made possible by the vital increase in the count of digital circuitry, its robust performance, comparatively lower cost, and its speed. If the number of bits increases the speed of operation is also increased.

It is nothing but the transistor acting as switch which nothing but a digital switch with two states. Hence, the digital information is referred as binary. Coming to reality the digital and binary are two auto correlated word which represents the kind of work required in digital circuits runs automatically on above based system.

Binary Logic States

Table 1 represents the logic table of different states which works on TTL Compatible logic of 0 to 5 volts divided as 0 to 1 volt in low and 2.5 to 5 volt in high logic state Volts. Below table also clear the idea of Boolean logic with its two different states.

Table 1: *Logic Table of 2 states*

Boolean Logic	Boolean Algebra	Voltage State(Positive)	Voltage State(Negative)
TRUE	1	High	Low
FALSE	0	Low	High

Figure 1 explains above logic in more detail which is represented as a simple switch works on following principles.

The labeled voltage is High (Low) when the label's stated function is True (False).
When switch is open then it is true states. (Recall that with the switch open, Ohm's Law implies that with zero current, the voltage difference across the "pull up" resistor is zero, so that the labeled point is at +5 Volts. With a closed switch, the labeled point is connected to ground, with a 5 Volt drop across the resistor and a current of $I = V = R = 5$ mA through it.)

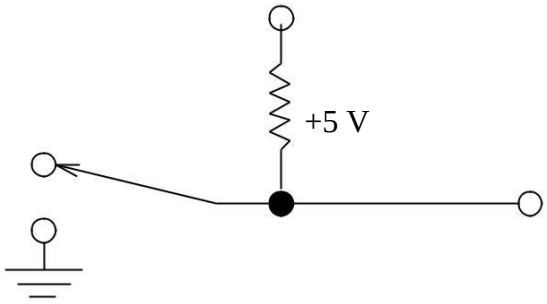

Fig 1: *Illustration for labeling logic states ("positive true").*

For negative state it is negative true where switch will be closed with a bar over it. Our statement reconstructed as below:
The labeled voltage is Low (High) when the label's stated function is True (False).
So in the figure, the stated function (switch closed) is true when the voltage is low.

Binary Arithmetic

Binary means two having two types of bit 1 or 0 .Bit is signal used in digital as an input and output at different digital device. There are many common conventions for representation of numbers in binary.
The most familiar is unsigned binary. An example of a 8-bit number in this case is

$(01001111)_2 = 0*2^7 + 1*2^6 + + 1*2^0 = 64 + 8 + 4 + 2 + 1 = (79)_{10}$

(Generally the subscripts will be omitted, since it will be clear from the context.) To convert from base 10 to binary, one can use a decomposition like above, or use the following algorithm illustrated by 79: 79=2 = 39, remainder 1, then 39=2 = 19 r 1, and so forth. Then assemble all the remainders in reverse order.

The largest number which can be represented by n bits is $2^n - 1$. For example, with 4 bits the largest number is $1111_2 = 15$.

The most significant bit (MSB) is the bit representing the highest power of 2, and the LSB represents the lowest power of 2.
Arithmetic with unsigned binary is analogous to decimal. For example 1-bit addition and multiplication are as follows: 0 + 0 = 0, 0 + 1 = 1, 1 + 1 = 0, 0 0 = 0, 0 1 = 0, and 1 1 = 1. Note that this is different from Boolean algebra, as we shall see shortly, where 1 + 1 = 1.
Another convention is called BCD (binary coded decimal). In this case each decimal digit is separately converted to binary.

Therefore, since $7 = 0111_2$ and $9 = 1001_2$, then 79 = 01111001 (BCD).as it is different than our previous result. We will use BCD quite often in this example .we can also convert it in gray code which provides the most of information of digital signal.

Hexadecimal Representation

Hexadecimal number system is having base 16 and can be represented by using four input bits from 0000 to 1111 .out of 16 numbers first nine value are represented as number 0 to 9 and rest as A to E.

Table 2: *Representation of Hex Number*

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E

Representation of Negative Numbers

There are two commonly used nomenclatures for representing negative numbers. Sign flag is set to high to represent the negative number. So, for example with 4-bit numbers we would have 0011 = 3 and 1011 = −3. This is simple to see, but is not good for doing arithmetic.

General way of designing negative number is2's complement, negative numbers are choose because it provides sum of a number and its 2's complement is zero. Consider the 4-bit example again, we have 0111 = 7 and its 2's complement −7 = 1001. Adding (remember to carry) gives 10000 = 0. Both addition and multiplication work as we have discussed using 2's complement. There are two methods for forming the 2's complement:

1. Make the transformation 0 *to* 1 and 1 *to* 0, then add 1.
2. Add some number to -2^{MSB} to get the number you want. For 4-bit numbers an example of finding the 2's complement of 7 is −7 = −8 + 1 = 1000 + 0001 = 1001.

Logic Gates and Combinational Logic

Gate Types and Truth Tables

The basic logic gates are AND, OR, NAND, NOR, XOR, INV, and BUF. The last two are not standard terms; they stand for inverter" and “buffer", respectively. The symbols for these gates and their corresponding Truth table are given in Table 3Every logical gate functions, as well as the Boolean output discussed in the next section, follow from the truth tables for the AND and OR gates.

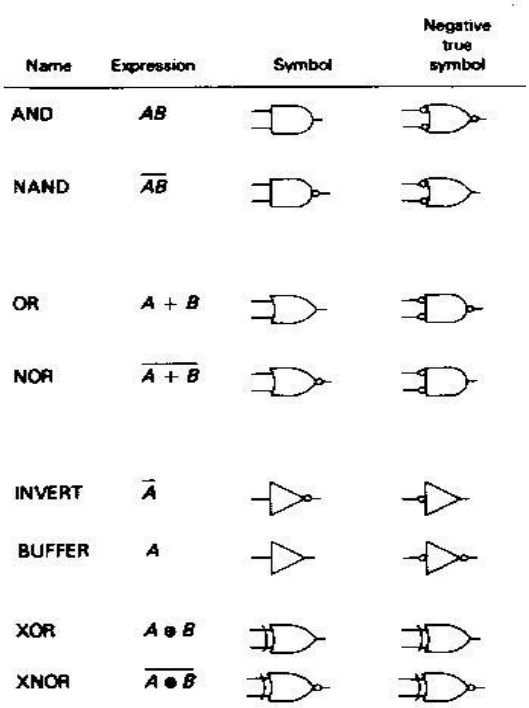

Fig 2: *Symbol for Logic Gates*

Table 3: *Truth Table for Logic Gates*

AND GATE		
A	B	Q
0	0	0
1	0	0
0	1	0
1	1	1

OR GATE		
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	1

XOR GATE		
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	0

Logical Instruction Set of 8085
With the help of truth table we can understand different instruction like

A) AND Gate
1. ANA R:
This instruction is used to perform logical AND operation .the contents of accumulator perform AND operation with specified register R and store result back to Accumulator.
e.g ANA B
If A=$(AA)_H$ & B=$(OF)_H$
1010 1010
0000 1111
0000 1010 =$(OA)_H$

ANA M
This instruction is used to perform logical AND operation .the contents of accumulator perform AND operation with Memory pointed by HL Pair and store result back to Accumulator.
e.g ANA M
If A=$(AA)_H$ & M=HL=$(1020)_H$=$(OF)_H$
1010 1010
0000 1111
0000 1010 =$(OA)_H$

ANIData(8)
This instruction is used to perform immediate logical AND operation .the contents of accumulator perform AND operation with data given within the instruction and store result back to Accumulator.
e.g ANI $(OF)_H$

If A= $(AA)_H$& TR=$(OF)_H$
1010 1010
0000 1111
0000 1010 =$(OA)_H$

B) OR Gate
ORA R
This instruction is used to perform logical OR operation .the contents of accumulator perform OR operation with specified register R and store result back to Accumulator.
e.g ORA B
If A=$(AA)_H$ & B=$(OF)_H$

1010 1010
0000 1111
1010 1111 =$(AF)_H$

ORA M
This instruction is used to perform logical OR operation .the contents of accumulator perform OR operation with Memory pointed by HL Pair and store result back to Accumulator.
e.g ORA M
If A=$(AA)_H$ & M=HL=$(1020)_H$=$(OF)_H$

1010 1010
0000 1111
1010 1111 =$(AF)_H$

ORI Data (8 bit)
This instruction is used to perform immediate logical OR operation .the contents of accumulator perform OR operation with data given within the instruction and store result back to Accumulator.
e.g ORI $(OF)_H$
If A= $(AA)_H$& TR=$(OF)_H$
1010 1010
0000 1111
1010 1111 =$(AF)_H$

B) X-OR Gate
XRA R
This instruction is used to perform logical XOR operation .the contents of accumulator perform XOR operation with specified register R and store result back to Accumulator.
e.g XRA B
If A=$(AA)_H$ & B=$(OF)_H$

1010 1010
0000 1111
1010 0101 =$(A5)_H$

XRA M

This instruction is used to perform logical XOR operation .the contents of accumulator perform XOR operation with Memory pointed by HL Pair and store result back to Accumulator.

e.g XRA M

If A=$(AA)_H$ & M=HL=$(1020)_H$=$(OF)_H$

1010 1010
0000 1111
1010 1111 =$(A5)_H$

XRI Data (8 bit)

This instruction is used to perform immediate logical XOR operation .the contents of accumulator perform XOR operation with data given within the instruction and store result back to Accumulator.

e.g XRI $(OF)_H$

If A= $(AA)_H$& TR=$(OF)_H$
1010 1010
0000 1111
1010 1111 =$(A5)_H$

CONCLUSION

In this paper, I have provided basic concept in designing of logical circuit using universal gates also to access the instruction set of 8085 microprocessor how digital electronics will be helpful also provided within this paper. Paper also inculcate the correlation between digital electronics and microprocessor .Flag register which is also going to be affected on the basis result of ALU which is nothing but an Flag Flip-flop in digital electronics. This is basic paper which provides fundamentals of digital electronics.

REFERENCES

1. Bennett, C.H., "Logical reversibility of Computation", IBM Journal of Research and Development, vol.17, pp. 525-532, 1973.
2. Vikram M. Kakade," Evolutionary Study and Performance Analysis of Generations in Wireless Technolgy", International Journal of Scientific & Engineering Research, Volume 4, Issue 6, June-2013.
3. Feynman, "Quantum mechanical computers", Optics News, pp. 11-20, 1985.
4. Vasudevan.D.P ,Lala.P.K and Parkesson.J.P, "Online Testable Reversible Logic circuit Design using NAND Blocks", Proc. Symposium on Defect and Fault Tolerance, pp. 324-331, 2004.
5. E. fredkin, T. Toffoli, "Conservative Logic", International Journal of Theory of Physics, vol.21, pp 219-253,1982.
6. Bhagyalakshmi, H.R. ;.Venkatesha, M.K, "An improved design of a multiplier using reversible logic gates", International Journal of Engineering Science and Technology Vol. 2,pp. 3838-3845,2010.
7. Carlin Vieri "Reversible Computing for Energy Efficient and Trustable computation", April 1998
8. Vikram M.Kakade, "OVERVIEW ON APPROCHES OF IMAGE SEGMENTATION WITH IT'S ALGORITHM AND APPLICATIONS." International Journal Of Engineering And Computer Science ,Volume 2 Issue 6 June, 2013 Page No. 1975-1980.
9. R. Landauer, "Irreversibility and Heat Generation in the Computational Process" IBM Journal of Research and Development, vol.5, pp. 183-191, 1961.