The Evolution of the Digital Revolution: Basic Theory

An Online Continuing Education Course for Engineers

Course Number: IC-3010

Credit: 3 Hours / 3 PDH / 3 CPD

The Evolution of the Digital Revolution
Basic Theory

Table of Contents

Subject	Page #'s
1. Introduction	4
2. The Binary Number System	4 - 5
3. The Hexadecimal System	5 - 6
4. Binary Coded Decimal System	6
5. AND, OR, and NOT Logical Functions	7 - 9
6. NOR's and NANDS	9 - 10
7. Exclusive OR (XOR)	10 – 11
8. Multiplexors and De Multiplexors	11-13
9. Boolean Algebra	14-15
10. Half Adder	15
11. Full Adder	15-17
12. Digital Computer Memory	17-22
13. Computer Clocks or Timing Circuits	22-24
14. A Special Circuit, a Shift Register	24-25
15. Counters	25-26
16. Newer Computer Circuits	26-27
17. Conclusions	27

List of Illustrations and Tables

Figure Or Table #	Title	Page #
Table 2.1	Counting in Binary	4
Table 2.2	Counting in Octal, only 27 of 64 Possible Numbers are shown	5
Table 3.1	Using 4 Bits to Count from 0 to 15 in Decimal or 0 to F in Hex	6
Table 4.1	The BCD Numbering System Derived from Binary	6
Figure 5.1	Representation of an AND Logic Circuit and Its Truth Table	7
Figure 5.2	3 Input AND gate and Truth Table	7
Figure 5.3	Minterm Maps for 2, 3, and 4 Input AND Gates	7
Figure 5.4	Sketch Showing How a Binary Bit Can Be Either a Logic Element or a Number	8
Figure 5.5	Picture of the Symbol, Truth Table, and Minterm Map of a 2 Input OR Gate	8
Figure 5.6	Three Input OR Gate with Its Truth Table and Minterm Map	9
Figure 5.7	Diagram of an NOT Gate and Its Truth Table	9

Figure 6.1 Symbol, Truth Table and Minterm Map for a Two Input NAND Gate 9
Figure 6.2 Symbol, Truth Table, and Minterm Map for a Two Input NOR Gate 10

Table 7.1 Table Showing What Happens When Two 1 Bit Binary Numbers Are Added 10
Figure 7.1 XOR Circuit Made from AND's, OR's, and NOT's; and XOR Truth Table 10
Figure 8.1 Two to One Multiplexor 11
Figure 8.2 Four to One Multiplexor 12
Figure 8.3 Multiplexor Circuit to Pick One of Four Inputs 12
Figure 8.4 Schematic Representations of a 1 to 2 and a 1 to 4 De Multiplexor 13
Figure 8.5 A way to make a De Multiplexor 13
Figure 9.1 Some of the Rules of Boolean Algebra 13
Figure 9.2 Truth Table and Minterm Map for DeMorgan's Theorems 14
Figure 9.3 A Modification of DeMorgan's Theorem 14
Figure 10.1 HALF ADDER Symbol, HALF ADDER Circuit, and HALF ADDER Truth Table 14
Figure 11.1 Symbol for a FULL ADDER and Its Truth Table 15
Figure 11.2 One Way to Implement a FULL ADDER 15
Figure 11.3 Classical Way to Implement a Full Adder with Two Half Adders and an OR 16
Figure 11.4 Truth Table for Figure 8.3 16
Figure 12.1 Picture of Donut Shaped Magnetic Core Showing How It Can Be Magnetized in Two Directions 17
Figure 12.2 Magnetic Core Showing X and Y Current Pulse Wires and Sense Wire 17
Figure 12.3 Simple Transistor Flip Flop 18
Figure 12.4 Symbol for a JK Flip Flop and Its Truth Table 19
Figure 12.5 Symbol, Truth Table, and Timing for a T or Toggle Flip Flop 19
Figure 12.6 Symbol for and Truth Table of a Set Reset (SR) Flip Flop 20
Figure 12.7 Symbol for and Truth Table of a D Flip Flop 20
Figure 13.1 Square Wave Signal That Can Be Used As a Computer Timing Signal 22
Figure 13.2 NE555 Timer Used As a Square Wave Oscillator 22
Figure 13.3 Schematic of a Pierce Crystal Controlled Oscillator 23
Figure 14.1 Serial In Parallel Out Shift Register Showing Operation 24
Figure 15.1 Presentation of an Eight Bit Binary Counter 25
Figure 15.2 A Three Bit Ripple Counter 25
Figure 15.3 Truth Table for a Toggle Flip Flop 26
Figure 15.4 Timing Diagram Showing How a Counter Built from Toggle Flip Flops Work 26
Figure 15.5 Three Bit Synchronous Counter Built with Clocked Toggle Flip Flops 27
Figure 15.6 Timing Diagram and Counting Chart for the Counter of Figure 15.5 28
Figure 16.1 Chart of Number of Transistors Versus Date Showing Moore's Law 28

1. Introduction

The digital computer has had an amazing impact on our way of life. Starting with first computers in the early 1940's to the present time is less than one century. In this less than n100 year span, we have gone from electrical mechanical marvels and vacuum tubes to electronic solid state devices with very few moving parts that operate almost flawlessly. Where computer mean time between failures was measured in days or even hours, a standard desk top or lap top computer is expected to run for years with no time off for good behavior, or maybe I should call it bad behavior. My own desktop has run almost continuously for 8 years, with the only failure being one of the output devices, the screen. It was replaced by an out of the box, plug in replacement for less than $200. It runs on 20% of the power of the one that burned out. So, to get on with the subject, some basic theory, let's proceed.

2. The Binary Number System

No series on digital computers would be complete without a review of the binary number system. The binary number system has only two digits, 1 and 0. This is very convenient because there are many electrical devices and circuits devices that only have two states. A switch can be on or off. We can let off be 0 and on be 1, or vice versa. A simple transistor circuit can be turned on or off. The on and off states can be thought of as a presence of some voltage (call this 1) and zero volts (call this 0).

The first thing that we need to do is to count in binary. Look at Table # 2.1.

2^2	2^1	2^0	Decimal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Table 2.1 Counting in Binary

The decimal number is equal to the binary digit in the 2^0 place (0 or 1) times 1 (which is 2^0) plus the binary digit in the 2^1 place times 2 (which is 2^1) plus the binary digit in the 2^2place times 4 (which is 2^2). The binary number needs more bits than three if we want to count higher than 7. Notice that counting to 7 is actually 8 positions, since 0 counts as part of the numbering sequence. As a matter of interest, we can take groups of 3 binary digits, as shown, and call it an octal numbering system. Table 1.2 shows this. The column on the left is counted as that digit times 8^1 (or 8). The column in the center is counted as that digit times 8^0 (or 1). Since showing all of the combinations would give a table counting to 77 octal (base 8), or 63 decimal (base 10) numbers, only 27 of 64 possible numbers are given. For fun, you might want to list all of the possible octal numbers from 00 to 77 and show their binary and decimal equivalents. One way to do this is to count from 000 000 to 111 111 in binary. Then list the octal and decimal equivalents.

Octal 8^1	Octal 8^0	Decimal	Octal 8^1	Octal 8^0	Decimal	Octal 8^1	Octal 8^0	Decimal	Octal 8^1	Octal 8^0	Decimal
0	0	0	1	0	8	0	0	0	0	0	0
0	1	1	1	1	9	1	0	8	1	1	9
0	2	2	1	2	10	2	0	16	2	2	18
0	3	3	1	3	11	3	0	24	3	3	27
0	4	4	1	4	12	4	0	32	4	4	36
0	5	5	1	5	13	5	0				45
0	6	6	1	6							54
0	7										63

wn

There are pens is that groups of 3 binary (check this out), we use 33_8 , w 33_8 than 011011_2. When work ven a code name to simplify wh tem of numbering is still used hat it is easier to keep track o numbering system is cal

3. The H

Whereas the c tem uses 4 bits and has 16 pos byte system of memory organ tal instead of binary because r of fact hexadecimal is hexadecimal. For example, a 3 hexadecimal or 15. To keep the number of c rs 10 to 15 are usually give f 15 (F) is still 1111.

Table 3.1 shows allow using only one character to denote each Hexadecimal nu replaced by the letters A to F. As an example of how this notation is used, imagine the 32 bit digital number (0101 0001 1101 0111). Looking at that string of 0's and1's can easily confuse most people. If we broke that 32 bit number into 4 groups of 4 bits each, and used Hexadecimal notation, this could be read as (51D7). Check this out by referring to Chart 3.1. Admittedly, this is still a bit cryptic, but it is definitely easier to read than 0101000111010111. The reason for learning to think in terms of binary, octal, and hexadecimal is that most modern computers are based on a binary system. IBM compatible machines have a memory that is based on 8 bit bytes. If we want to look at what is in those 8 bit bytes, a single character, 0 to F is easier than a 4 bit string of 0's and 1's. We will now look at a special form of 4 bit groups called Binary Coded Decimal.

