David Thompson

Information Technology 124 Reference Sheet

Here is my revised list of glossary terms. I reorganized all of them to match more closely with the course text book. Terms with a definition in ***bolded italics*** are ones I added and they need to be vetted.

**Supplemental 1**

**Voltage (E): *Named after Alessandro Volta, Voltage (E) is the force that causes electrons to want to move, and thus often also referred to as electro-motive force (EMF). Can be produced by batteries, rubbing, and by generators. Measured in volts (V)***

**Current (I): *The flow of electrons. Measured in Amperes (A), named after André Marie Ampére. Often shortened to Amps. 1 Amp = 1 Coulomb/sec, where 1 Coulomb ≈ 6.241x1018 electrons.***

**Resistance (R): *Opposition to electron flow (current). Measured in Ohms (Ω), named after Georg Ohm.***

**Power (P): *Energy per unit time. The product of the current (I) and the voltage (E). Measured in Watts (W). 1W = 1 Joule/sec***

**Frequency (f): *Cycles per second. Measured in Hertz (Hz). Prior to 1960, the units were cps (cycles per second).***

**Ohm’s Law: *The amount of current that flows is directly proportional to the voltage, and inversely proportional to the resistance. (I = E/R)***

**Ampacity*: The maximum current that an electric power source can deliver. For example, in batteries, the larger the surface area of a battery, the greater the ampacity and the lower the internal resistance of that battery.***

**Parallel Circuit: *A circuit which comprises branches so that the current divides and only part of it flows through any branch. All elements in the circuit have the same voltage drop, and each resistor drops all of the total voltage.***

**Series Circuit: *A circuit which comprises a path along which the whole current flows through each component. Each resistor drops only a portion of the total voltage.***

**Supplemental 2**

**Precision: *Another word for repeatability. Measured in percentage or ppm; a low value is desired.***

**Accuracy: *Defined as how close a given measurement is to the known or actual value. Measured in percentage or ppm; a low value is desired.***

**Resolution: *The resolution of a meter specifies the smallest part it can resolve, and is specified in %, ppm, ppb, or # of digits. Most common resolution is 1 part out of 2000, or 500 ppm, or 3½ digits.***

**Chapter 1 (Intro to Computers)**

**Turing Machine: *Designed by Alan Turing in 1937, a Turing machine is a mathematical model of a device that can perform any computation.***

**Analog vs. Digital**: The best example of this is in the clock differences. Analog means being progressive with infinitely small spaces between values (continuously variable), and digital means being variable in discrete increments, also being two states such as black and white, on or off, or yes or no.

**Positional Numbering Systems**: This refers to the digits of our counting system. In our decimal system we use (0-9), though it is common in IT to use Hexadecimal (0-F) and Binary (0-1). It refers specifically to the fact that the position of a digit is what gives it value, in addition to the value of the digit. For instance, in the number 3,693, the left 3 has 1,000 times more value than the 3 on the right, because of its position. Both binary and decimal are examples of positional numbering systems.

**Bit**: A “binary digit”. This is the smallest unit of data in a computer. It is represented in binary (0-1).

**Nibble**: Half of a byte, or 4 Bits.

**Byte**: A block of information made up of 8 Bits.

**Most Significant Bit**: The binary digit furthest to the left. So called because it creates the biggest difference to the value, since binary (like decimal) is also a positional numbering system.

**Least Significant Bit**: The binary digit to the furthest right. So called because it will only change the end value by 1, the smallest integer amount possible to change a number.

**Chapter 2 (Bits, Data Types, and Operations)**

**Data Type: *a representation of data as well as the operations that can be performed on the data. Examples: Unsigned Integers, Signed Integers, Real Numbers, Text characters, Booleans, BCD, ASCII, binary, floating point.***

**BCD**: Binary-Coded Decimal. This is another way to represent numbers. This method is commonly used in conversions between Hex and binary and functions based on 4 bits representing 1 decimal number.

**Signed Numbers**: Binary numbers with values giving specifically positive or negative signs. These must be specifically stated, otherwise we cannot be sure what signs they have.

**Overflow**: Values that are outside of the viewable set of numbers. Can occur when shifting, adding, or multiplying normally. ***If the sum of two binary numbers uses more bits than what has been allotted, overflow occurs, and the MSB is carried out, resulting in an incorrect sum***

**Signed Magnitude**: Converting the position of the MSB to a sign placeholder, 1 being negative (-) and 0 being positive (+).

**2’s Complement**: A method of representing negative numbers, achieved by inverting all of the bits in a number and adding 1 to the result. This method is only for representing negative numbers; positive numbers are expressed normally.

**ASCII**: American Standard Code for Information Interchange. ASCII code is used to represent text including punctuation, letters, numbers and accent marks. ASCII uses a 7-bit code, or Hex for each specific character. ASCII is now a subset of Unicode.

**Truth Tables**: Used to study Boolean Logic as well as in exhaustive proofs. A table usually with two sides, with the left side being inputs and the right side being outputs. Used in K-Maps and logic gate schematics as well.

**Unicode:** The original Unicode Standard used 2 bytes/character, and was thus able to represent 65,536 characters. It includes 7-bit ASCII as a subset. The most recent version of Unicode uses 18 bits, but only about 144,000 of the possible 262,144 (218) characters are presently defined.

**Boolean Logic**: A study of formal logic, focusing on true and false (two values) logic. There are many names for the pieces of logic (yes/no, on/off, 1/0). Equations will use Boolean Operators and variables to make true statements. There is an associated set of theorems and identities used to help minimize the length and redundancy of Boolean expressions.

**Boolean Operators**: AND, OR, NOT

**AND:** *C is true only if A and B are also true. Can also be thought of as a series circuit.*

**OR:** *C is true if either A or B is true. Can also be thought of as a parallel circuit.*

**NOT:** *returns the opposite of whatever value it is given. Also called an inverter.*

**NAND:** *C is false only when A, B are both false.*

**NOR:** C is false *when either A or B is true.*

**XOR:** C is only true if either A or B is true, but not both.

**XNOR:** C is only false if either A or B is true, but not both.

**DeMorgan’s Law:** *If we NOT two input values (A and B), AND them, and then NOT the result, we get the same result as an OR operation. The same is true going in reverse: if we NOT two input values (A and B), OR them, and then NOT the result, we get the same result as an AND operation.*

**Terms:** Variables in a Boolean expression or equation which represent an input or output (i.e. A, B, F)

**Sum of Products (SOP)**: refers to two-level logic in which the terms are first products (ANDed), then the products are summed (ORed) together. An example would be: F=AB’+A’B

**Product of Sums (POS**): refers to two-level logic in which the terms are first sums (ORed), then the sums are multiplied (ANDed) together. An example would be: F’=(A’+B’)+(A+B)

**Minterm Expansion**: The SOP way of writing the minimal expression for a Boolean equation. This should consist of all terms that are true.

**Maxterm Expansion**: The POS way of expressing a minimal expression in a Boolean equation. These terms will be all of the false outputs for the given inputs.

**Boolean Minimal Expressions**: These are equations with the fewest necessary terms and will either be in minterm or maxterm form. The opposite of the equation will be the opposite form of expressing the equation.

**Bubble Matching**: A strategy for complying to a logic set that limits what types of gates you can use and how they are depicted in the schematic. One example would be using NOR - Invert-AND (NOR – NOR) instead of an OR-AND.

**Grey Code**: Code in which each new binary group is only one bit different from the last. This is used in K-Maps.

**Delay:** The time from when a change enters a logic gate until that change propagates through the gate to the output.

**Critical Path:** There are often multiple paths through the transistors of a logic gate; the path that takes the longest in a given condition is the critical path.

**Input glitches:** Momentary and undesired transitions in the data on the input to a logic gate or storage element.

**Chapter 3 (Digital Logic Structures)**

**Transistors**: The building blocks of gates. These are pieces of hardware that can pass current or block current. These are used in creating logic gates which implement the Boolean operators. Logically, each transistor acts as a voltage-controlled switch

**P-type Transistor:** In CMOS, it is the *transistor whose source is connected to +3.0V. When the gate is 1, the transistor is off. When the gate is 0, the transistor is on****.***

**N-type Transistor:** In CMOS, it is the transistor whose drain is connected to 0V. When the gate is 1, the transistor is on. When the gate is 0, the transistor is off.

**Gates**: These use Boolean logic in order to create desired outputs. They come in several different forms, though we deal primarily with AND, OR, NOT, NAND, NOR and XOR. Normally one gate will have 1-4 inputs, with a single output.

**Counters:** A circuit that counts through a predefined sequence. Uses flip flops or latches to hold current state, and uses input forming logic to predict its next state. Outputs are clocked.

**Multilevel logic**: This is referring to the amount of layers in a schematic design. Example: 2 NAND gates feeding into an OR gate would be 2-level logic.

**Schematic**: In IT this will almost always refer to a diagram of a group of connected gates, multiplexers, counters… etc., showing how they are connected.

**Functionally Complete Set**: A collection of gates that are needed to implement any set of logic. NAND and NOR gates, for example, can perform any sort of logic and therefore constitute a functionally complete set by themselves.

**Combinational Logic**: A circuit which requires only its current state inputs to execute a function.

**Sequential Logic**: A circuit which requires both current state and former state inputs to execute a function. Requires storage element (memory).

**Multiplexer**: (MUX) A circuit with at least one input being a selector, allowing the output to be connected to the selected input.

**Decoders**: These work in the opposite way to mux circuits. Decoders will take a few inputs and output more. Such as inputting 2, and outputting 4.

**Read Only Memory (ROM)**: A type of memory that stores data that can only be read but not altered. Commonly used for Lookup Tables.

**Adder**: At the bit level, there are quarter adders, half adders and full adders. At the multi-bit level, there are ripple adders and look-ahead adders, all of which combine various binary adders together. Ripple adders and look-ahead adders will carry the values through multiple bit-level adders. Adders can be used as subtractors if they add negative numbers.

**Latches:** A level-sensitive digital storage element. There are many different variations such as SR, Gated, JK, T, D, and Master/Slave. Latches will respond to their inputs, and usually provide both the true and the complement of the output.

**Flip-Flops:** (FF) An edge-sensitive digital storage element. Used in computer registers and sequential logic, and made with gates and transistors.

**Rising/ Falling Edge:** A term referring to the clock of a sequential logic storage element such as a flip-flop or a latch, simply meaning the change when the clock goes high or low, being rising and falling accordingly.

**Address Space:** The number of locations in memory, usually a power of 2.

**Addressability:** Number of bits per memory location.

**Register:** Storage (one or more) used to hold a value in a computer or microprocessor. Variations include instruction registers, address registers, data registers, and the program counter. Registers are made up of flip flops or latches, which are the fastest storage elements.

**State:**The state of a system is a snapshot of all the relevant data that describes the system.

**State Diagram:** Shows all states (nodes), output value for each state (value in node), transitions (edges) between states for input values (edge labels).

**State Graph:** A visual representation of the flow and conditions of a state machine. These can include inputs and outputs, normally accompanied by a truth table.

**(Finite) State Machine:** An abstract model for circuitry that changes its states as a function of the present state and the inputs, and which provides outputs which correspond to their respective states.

**KMaps**: Karnaugh Maps let us represent the inputs and outputs of a truth table in a way that allows us to see the patterns. By placing the outputs within the squares, you can more easily determine the minimal expression of an equation. The inputs are written in grey code, meaning that the positions (which are used in Min/Maxterm expressions) will not be in an easily predictable order.

**Don’t Cares**: These are outputs in KMaps that do not need to be true or false in order to function. They are included with true outputs when convenient, though they are not necessary.

**Chapter 4 (von Neumann Model)**

**Von Neumann Model:** The basic computer model first proposed in the 1940’s. It consists of five parts: memory, a processing unit, input, output, and a control unit.

**Instruction Cycle:** Not to be confused with the clock cycle, the instruction cycle is the sequence of six steps required to process an instruction. The steps are fetch, decode, evaluate address, fetch operands, execute, and store result.

**Chapter 5-9 (programming)**

**Recursion:** a technique in which a function can call itself.

**Stack:** A last-in first-out (LIFO) data structure***.***

**Queue:** A first-in first-out (FIFO) data structure.

**Label:** A symbolic name used to identify memory locations that are referred to explicitly in the program.

**Subroutine:** A portion of code that can be invoked multiple times within the same program without having to specify its details in the source program each time it is used. Also called procedures or functions.

**Privilege:** The right to do something, such as execute a particular instruction or access a particular memory location.

**Priority:**The urgency of a program to execute. Each program is assigned a priority, specifying its urgency compared to all other programs.