Grade 11 Computer Engineering Review

# Computer Hardware/Software

**ATX** – Power supply standard for PCs

**BIOS** – Basic Input Output System. Checking RAM. Setting up boot drive

# Networking

**Network Card** – PCs physical connection to network: Ethernet or WiFi

**Switch** – LAN Technology (local area network). Ethernet or WiFi. Local communication only

**Ethernet Cable** – Physical connection to network

**Router** – WAN technology (wide area network or Internet). Ethernet on the LAN side. Different technologies on the WAN side: Serial Modem (56K), DSL, Cable Modem, 3G, LTE, Satellite

**Default Gateway** – AKA the router. When configuring your computer to access the WAN internet cloud, you need to specify the IP address of your router.

**Subnet Mask** – Is used to filter IP addresses using AND logic, the IP address and Subnet mask are calculated and the result will determine if you can communicate.

**LAN VS WAN** – LAN 🡪 Local Traffic (IP Address destinations that are similar to your own). WAN 🡪 Traffic that does not belong to your local network. The router (default gateway) will determine where the traffic goes.

**Ping** – send and receive protocol to check if you can reach a destination on a LAN or WAN. Ping also shows you how long it takes to reach a destination.

**Tracert** – command that shows you the path that your packets take to get to its destination. Path through routers.

**Ipconfig** – shows you your current TCP/IP configuration. “piconfig/all” shows even more stuff

**Cloud** – The routers and servers that make up the internet. There are many different paths that you can take through the cloud. There are many different servers available on the cloud: web servers, ftp servers, game servers, iCloud servers, etc, etc… you can even create your own server on the cloud.

**Client Server** – ALL internet applications are client/server. When you surf the web using Chrome, you are using a web client connecting to a web server. When you play Warcraft, you are using a game client connecting to a game server.

**Port Numbers** – Every networked computer has many “ports” into the system. Each port is like a different doorway into your computer. Different network applications run on different port numbers. Some popular port numbers are:

* 80 – http web servers and clients
* 8080 – used for proxy servers and sometimes web servers.
* 21 – ftp servers and clients. Old school file transfer protocol before web servers.
* 23 – telnet servers and clients. Old school command line remote control protocol.
* 6112 – Battlenet server and client. All Blizzard games communicate through this port.

Different servers use different port numbers... You cannot run two dedicated servers on listening to one port.

Open “ports” in your computer might invite hackers into your system. They may try to find an open port, send “bad” data to the port, causing your system to crash, or even worse, inject remote control code turning your computer into a zombie. The best way to close your ports is to run firewall software. If you have to open a port for a game, or your own personal web server, make sure your software is patched to the latest version.

**IPv4 Addresses** – IP Addresses are the address of your computer on the network. Your IP address has 2 parts: 1. “Area Code”, 2. Computer. When a packet is sent over a network, the area code is first checked. If the area code is local, the packet stays in the switch, and goes to the local computer. If the area code is foreign, the packet is sent to the default gateway and the router sends the packet to the appropriate area code.

North Side Area Code: 70.X.X.X

South Side Area Code: 150.150.X.X

**Servers** – When talking about network servers, there are two modes of thinking.

1. Physical Hardware Servers – there are machines in the cloud with CPU/RAM/Hard Drives that are connected to the cloud at a specific IP address.
2. Software Servers – The machines in the cloud run one or more server software running on different port numbers. Ex. staugustinechs.ca server. There is a physical server stored at the Netfirm office. This physical server is running 3 different software servers: http port 80, ftp port 21, MySQL server 3306. Ex2. Ycdsb.ca server. http 80, pop3 email 110, proxy 8080

# Logic Gates

Below are the 7 Basic Logic Gates

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Symbol** | **Equation** | **Tabular** |
| AND | If all inputs are true, output is true | C:\Users\Matthew\Desktop\100px-AND_ANSI.svg.png | Z = a\*b | |  |  |  | | --- | --- | --- | | **A** | **B** | **Z** | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | |
| OR | You need at least 1 input to be true for the output to be true | C:\Users\Matthew\Desktop\100px-OR_ANSI.svg.png | *Z = A* + *B* | |  |  |  | | --- | --- | --- | | **A** | **B** | **Z** | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | |
| XOR | You need one or the other but not both for the output to be true | C:\Users\Matthew\Desktop\100px-XOR_ANSI.svg.png | Z = A \oplus B | |  |  |  | | --- | --- | --- | | **A** | **B** | **Z** | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |
| NOT | Take your input and reverse the output | C:\Users\Matthew\Desktop\100px-NOT_ANSI.svg.png | Z = C:\Users\Matthew\Desktop\9c17a029f4d702734b1c45625e44f304.png | |  |  | | --- | --- | | **A** | **Z** | | 0 | 1 | | 1 | 1 | |
| NAND | Just like AND, except the output is opposite | C:\Users\Matthew\Desktop\100px-NAND_ANSI.svg.png | Z = C:\Users\Matthew\Desktop\b0d4cc86db2abbb94193ae2cb2688f8d.png | |  |  |  | | --- | --- | --- | | **A** | **B** | **Z** | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |
| NOR | Just like OR, except the output is opposite | C:\Users\Matthew\Desktop\100px-NOR_ANSI.svg.png | Z = C:\Users\Matthew\Desktop\693b17d93f89478d618c54e8574586ba.png | |  |  |  | | --- | --- | --- | | **A** | **B** | **Z** | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 0 | |
| XNOR | Just like XOR, except the output is opposite (AND) | C:\Users\Matthew\Desktop\100px-XNOR_ANSI.svg.png | Z = \overline{A \oplus B}or {A \odot B} | |  |  |  | | --- | --- | --- | | **A** | **B** | **Z** | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | |

# Binary and Hex Numbers

Here is an example conversion table:

|  |  |  |
| --- | --- | --- |
| **DEC** | **BIN** | **HEX** |
| 49 | 00110001 | 3 1 |
| 222 | 11011110 | D E |
| 175 | 10101111 | A F |
| 43 | 00101011 | 2 B |

# Binary Math

|  |  |  |
| --- | --- | --- |
| 01011011  +00100001  01111100 | 10011001  +01110011  100001100 | 11011111  +00111111  100011110 |
| 10111101  - 00110101  10001000 | 10011001  -01011001  10111111 | 10001100  -00011001  01100010 |
| 1001  \*1001  1001  01001000  01010001 | 1100  \*0011  1100  00011000  00100100 | 1101  \*0111  1101  00011010  00110100  01011011 |

# Converting Between Boolean Tables, Equations and Drawings

## Truth Tables to Diagrams

Logic that is used when starting a motor vehicle:

|  |  |  |  |
| --- | --- | --- | --- |
| **In Park?**  **A** | **In Neutral?**  **B** | **Key Turned?**  **C** | **Start the Car**  **Z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | ? |
| 1 | 1 | 1 | ?\* |

Steps

1. Highlight the rows that are on. Don’t worry about the 0 rows.
2. For each row that is highlighted, build an AND circuit representing the line that NOTing any input that is 0.
3. Connect the AND circuits together with an OR gate and you are done!

## Diagrams to Truth Tables

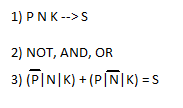
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **Z** | **D** | **E** |
| 0 | 0 | 0 | 1 | **0** | **1** |
| 0 | 0 | 1 | 0 | **0** | **0** |
| 0 | 1 | 0 | 1 | **0** | **1** |
| 1 | 0 | 0 | 1 | **0** | **0** |
| 0 | 1 | 1 | 1 | **0** | **0** |
| 1 | 0 | 1 | 0 | **1** | **0** |
| 1 | 1 | 0 | 1 | **1** | **0** |
| 1 | 1 | 1 | 1 | **1** | **1** |

Here are the steps:

1. Identify inputs and outputs
2. Create an empty table with headers that represent the inputs and outputs
3. Fill in the input columns with every possible combo
4. Create “midpoint” columns for every logic gate in the circuit
5. Start filling in the midpoint columns based on the logic gate until you get the output column
6. Done

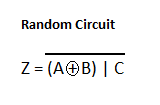
**This Writing = unneeded**

## Diagrams to Equations

Not everyone uses diagrams or tables. Mathematicians use equations. Below are the steps to convert a Boolean diagram to algebra.

1. Identify the Inputs and outputs
2. Identify the logic gate used in the diagram
3. Start building the equation logic gate by logic gate
4. Attach the equation logics using brackets to specify order of operations
5. Done!

## Equations to Diagrams

Here are the steps to convert Boolean equations to diagrams:

1. Identify the inputs and outputs
2. Identify the logic gates in the equation
3. Start building circuits for each logic gate
4. Connect the logic gates based on the bracket order of the equation
5. Done!

# Users of the Different Logic Representations

Truth Tables – Business People, Mathematicians, Statistician, Computer Engineers

Equations – Mathematicians, Scientists, Philosophers, Computer Engineers, Computer Scientists

Circuit Diagrams – Computer Engineers, Circuit Builders, Electrician, Electrical Engineers

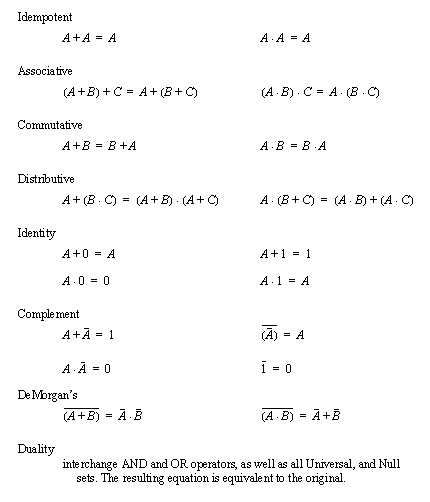
Circuit Datasheets – Computer Engineers, Circuit Builders, Electrician, Electrical Engineers

74LS00 (NAND Microchip)

# C:\Users\Matthew\Documents\Computer Engineering 2012\simplification.pngBoolean Algebra Simplification

When you get a complex Boolean equation, it is possible to use some Boolean identities to shrink that equation to a smaller size while retaining the exact same logic.

In this example (🡪), the equation started with 8 gates: 4 AND, 3 NOT and 1 OR. By the end of the simplification, the equation ended with 4 gates: 3 And, 1 OR.



Example:

K[(P.N+P.N) = S

K.(P + N) = S

Example 2:

Z = ((A.C).B)+(A.C)

Z = (A.C).(B+1)

Z = (A.C).1

Z = (A.C)