**Name - Srajan Ahuja**

**100 number - 1001858728**

**Date - 03/06/2021**

**Homework 3**

**Q1. Design a “Half Adder” circuit for two inputs A and B. Now, design a “Full Adder”**

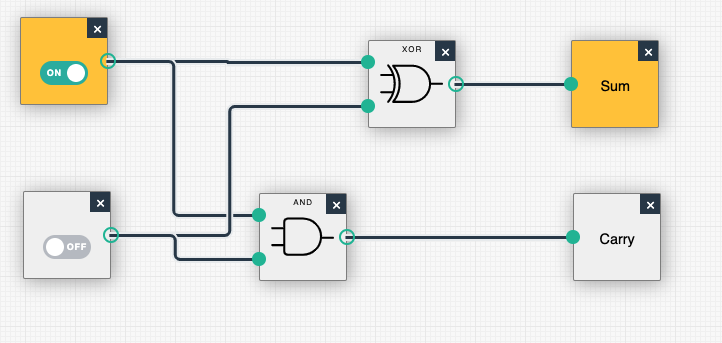
**circuit using previously designed half adders.**

**[15 points]**

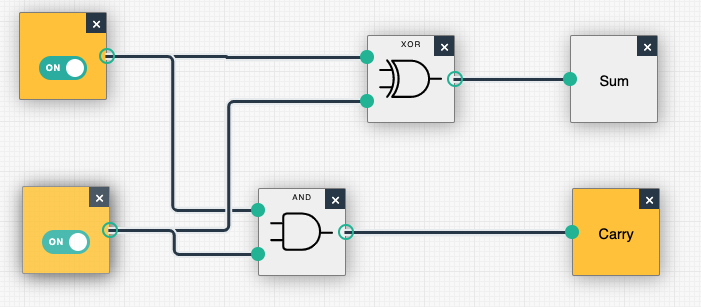
A1. The circuits are below :-

1. **Half Adder :-** XOR takes care of the binary addition, while AND checks for carry bit.

**A = 1, B = 0 , Sum = 1, Carry = 0**

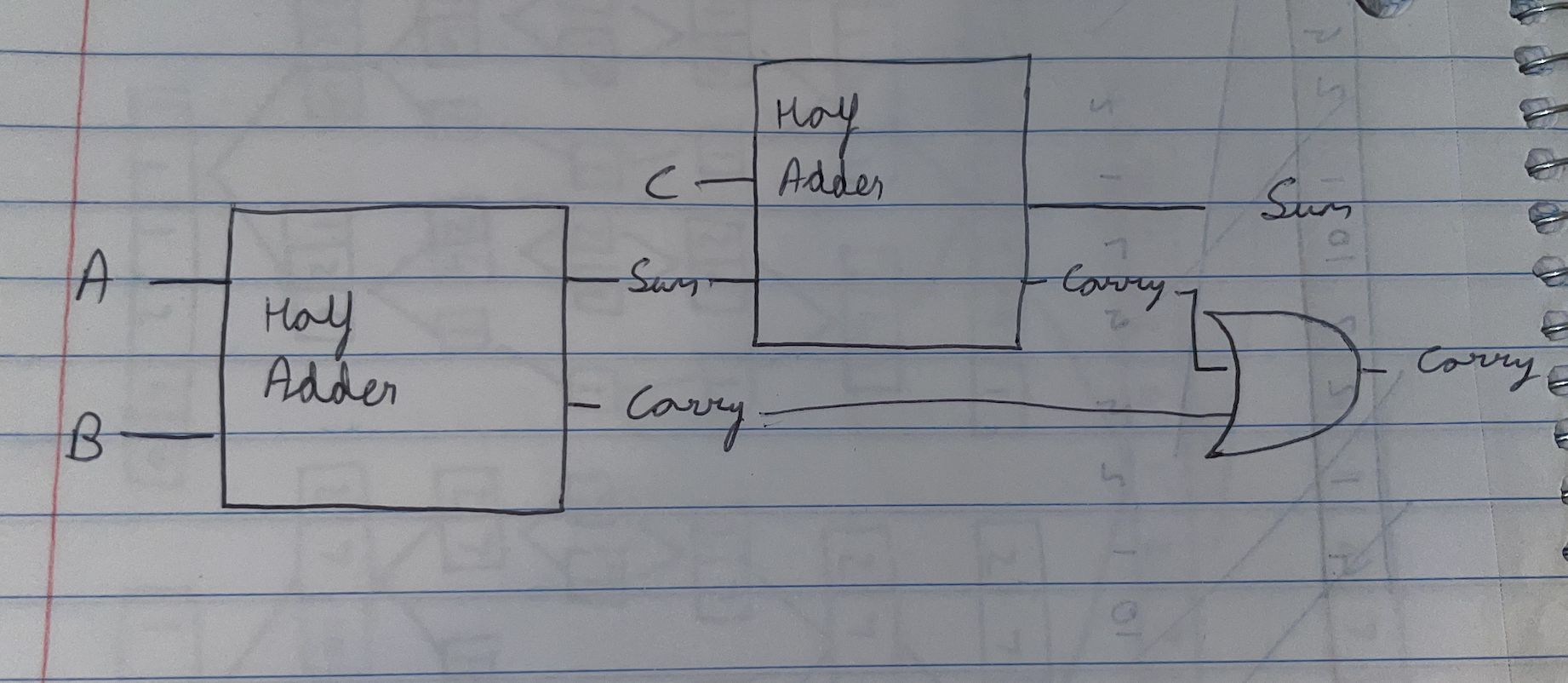


**A = 1, B = 1 , Sum = 0, Carry = 1**

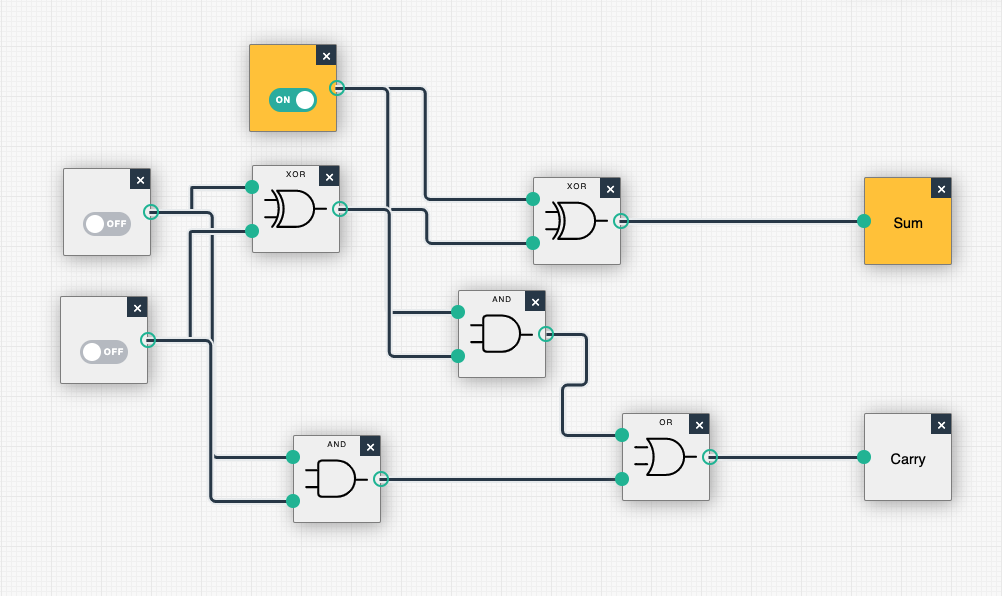


1. **Full Adder :-** Sum bit from Half Adder of A and B goes into a Half Adder with C, if either one of the half adders has a carry bit, the carry bit output becomes 1 (OR gate).

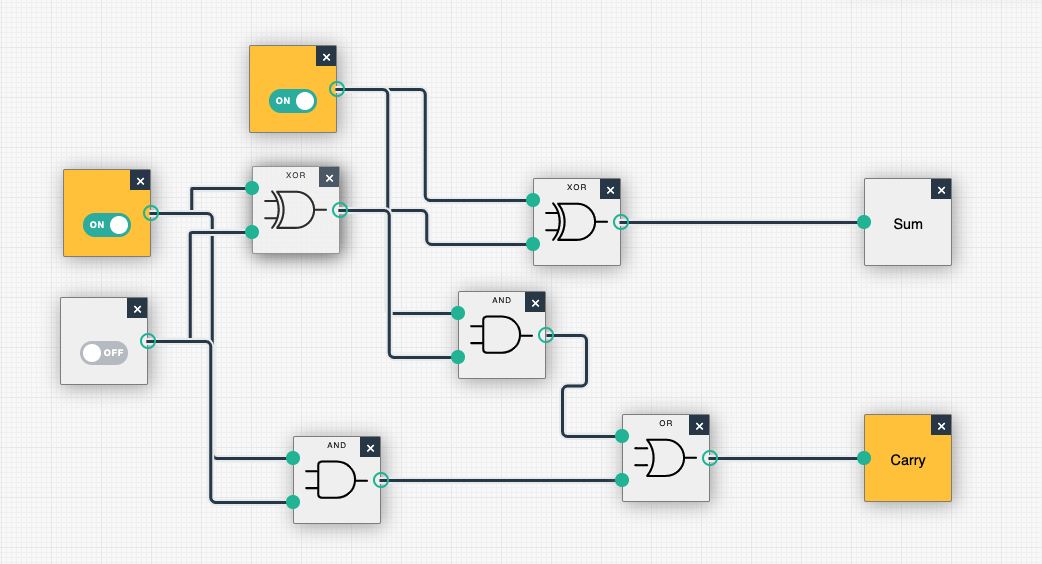
**Logic**



**A = 0, B = 0, C = 1, Sum = 1, Carry = 0**



**A = 1, B = 0, C = 1, Sum = 0, Carry = 1**



[Simulations created using Logic gate simulator on academo.org]

**Q2.What is an overflow? When does it occur?[6 points]**

A2. If there is an attempt to use or store any value that is more than the capacity of the CPU (or the word size) this causes the CPU to have insufficient bits to be able to use or store that number and it drops the extra bit, this is called overflow.

A common example of occurrence of overflow would be :-

Let’s say our CPU can store only 8 bits and we want to add 255 and 1. So in binary the addition would be like :-

255 1 1 1 1 1 1 1 1

* 1 + 0 0 0 0 0 0 0 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

256(expected answer) **1** 0 0 0 0 0 0 0 0

But since the CPU can’t store more than 8 bits, the 9th bit gets tossed out and we get :-

0(what the CPU stores) 0 0 0 0 0 0 0 0

**Q3. What is Random Access Memory? Why is this called so? [6 points]**

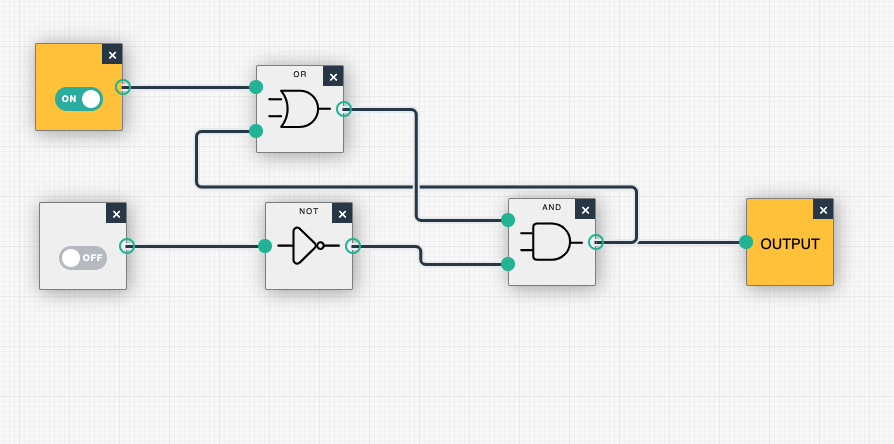
A3. Random Access Memory (RAM) in simple terms is a short term or working memory that a CPU uses while performing a task. It is used so that the CPU doesn’t have to go back to read data from the hard drive every time it is required to complete the task.

It is called so as the CPU can access any memory location directly, all it’d require would be a column number and row number to go to in the matrix without having to go through previous addresses.

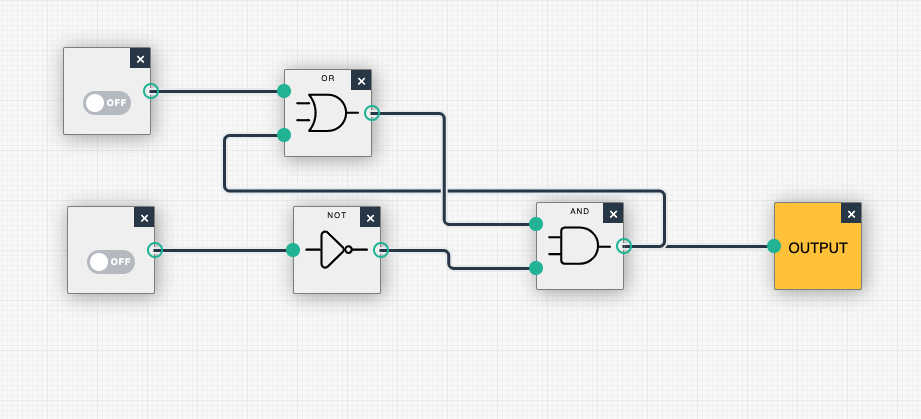
**Q4. What is “And-Or Latch”? Why is this called “latch”? [5 points]**

A4. The And-Or Latch is a combination of the AND, OR & NOT logic gates which is used to store a bit of data. It has two inputs: Data bit and Reset bit, the data bit is the bit to be stored while the reset bit determines whether or not the data bit(current one) is to be stored.

Data = 1, Reset = 0, this stores 1 in the output.



Now even if we change the Data to 0, it won’t change the output since we didn’t change the Reset bit to allow changes.



This is how it stores data and vice-versa for storing 0 instead of 1.

It is called a latch as in simple terms it latches onto the last input and stores it unless instructed otherwise.

**Q5. What is a “register”? [3 points]**

A5.Since a latch can only store a bit of information, in order to store bigger data (let’s say an 8 bit number), we’d put together 8 latches in a row and store 8 bits of the number in it. Such a combination/group of latches is called a register. Registers are used by the CPU to store addresses or instructions while executing a task.

**Q6.What is “multiplexer”?[3 points]**

A6. A multiplexer is a device which takes multiple inputs and gives the output through a single line. In Computers Multiplexer acts as a data selector, by taking in the address inputs from the cpu it determines the destination register it needs to send the data/signal to thus making traversing to the desired register easier for the computer.

**Q7.Explain fetch phase, decode phase, and execute phase in short.[12 points]**

A7.

1. Fetch Phase - This is the process in which the CPU is obtaining instructions that are stored in the RAM. Let’s say we turn on our PC, the program counter(value on the Instruction Address Register) gets set to 0 by default. So in this fetch phase, the value on the Instruction Address register is 0 and the CPU will fetch the instruction data present at the 0th address of the RAM and write it on the Instruction Register.
2. Decode Phase - Now that we have the instruction in our Instruction Register, the CPU needs to figure out what it is. In order to do this the instruction is split into two parts, the first part contains the opcode, the opcode tells the CPU what kind of an operation is to be performed and the second part of the instruction is the address in the ram of the data to be used in this operation.
3. Execute Phase - Now that the CPU knows what operation to perform and what data to perform it with, it executes the said instruction and increments the program counter by 1 for the CPU to move on to the next instruction in the RAM, this is known as the execute phase.

**Q8.What is clock speed? What is overclocking and underclocking?[6 points]**

A8.Clock speed is the rate at which a CPU can complete a fetch-decode-execute (processing) cycle. It’s unit is Hertz, where 1 Hertz means 1 cycle is completed by the CPU in 1 second.

1. Overclocking - It is when the clock speed of a CPU is increased to increase the processing rate which reduces the response time of the CPU significantly. It can cause the CPU to heat up which can in the worst case lead to the CPU producing garbage values or permanent damage to it’s components or both. Generally CPUs come with certain resistance to overclocking, so it’s okay if it’s done correctly.
2. Underclocking - It is when the clock speed of a CPU is decreased. The CPUs are designed to do the tasks from high difficulty (that may or may not require overclocking) to easy ones (that can be done on one-tenth of the maximum clock speed), so the CPU doesn’t always have to be on it’s maximum clock speed, if the task isn’t very demanding or let’s say the PC is idle then the CPU can underclock and hence a lot of power.

**Q9.Why is security important for Programming Language Design?[8 points]**

A9. Security plays an important role when it comes to language design. Let’s take the example of C vs Java to explain the importance of Security. Let’s say we have the same program that takes some user input and has 5 different functions to execute on that input. Here are the following security flaws:-

1. Type Checking - Type checking would be done in Java but not in C and this would help the programmer to rather not execute the program at all.
2. Pointers - Anyone using the application built in C can access the system/memory addresses since C uses pointers which compromises the security of the system. Whereas for the application built in Java, this won’t be the case as it restricts pointers.
3. Try-catch block - Java has a feature which enables the program to execute/choose an alternative execution plan in case a certain block(or in this case one of the functions in this application) doesn’t work. Whereas the C application won’t execute at all on encountering any errors.

In conclusion, having these security concerns in mind while designing a programming language makes the programming language semantically safe.

**Q10. What would be some of the design goals if you were to design a Programming Language? Explain your answer. [15 points]**

A10.

I would keep in mind the following design goals while designing a programming language:-

1. Generality - The lower the number of special use cases, the easier it is for anyone to use the programming language.
2. Uniformity - It should be uniform throughout, that is, if an integer is declared and a semicolon is used after that then a semicolon should be used after declaring a floating point number too.
3. Security - The language should have secure features like the code should be verified before execution so that we know if there are any flaws in the syntax or not and the language should not directly be giving out memory block addresses and rather interact with an intermediary first for security reasons.
4. Orthogonality - Two constructs can be incorporated in each other easily to be used in a meaningful way (for example using functions inside classes).
5. Readability - No programmer wants to read a code that's difficult to decipher or requires a huge learning curve to understand.
6. Writability - It should be easy to express for the programmer.

**Q11. What is the “writability” of a programming language? [3 points]**

A11.

Writability is the quality of a programming language that makes it easier for a programmer to express their logic/flow for their desired algorithm in a clear, correct and concise way.

**Q12. What are the differences between primitive type and reference type in Java? [5 points]**

A12.

1. Primitive types are predefined (int, float, boolean, etc.) whereas Reference types are created by the programmer (array, class, etc).
2. Primitive types store values corresponding to every variable whereas Reference types store the address in the memory that’s pointing to a heap memory containing the objects in it.
3. Primitive types can only store one type of value whereas Reference types can store multiple types of values.

**Q13. What is an “extra semicolon problem” in C++? [3 points]**

A13.

In order for C++ to be widely adapted by the existing programmers (when C was very popular), C++ was made very similar to C and a byproduct of this was that after declaring a class in C++ (an object) a semicolon was used but it wasn’t the case after declaring a function which caused irregularities and messed with the uniformity of C++.

**Q14. What is “macro”? Write a piece of code using a macro.[7 points]**

A14.

A macro is a small code that stands for a long piece of code, let’s say our code has a lot of repetitions in it then a macro would be a solution to create abstractions for that particular repetitive piece of code.

Though this is similar to functions but functions are compiled whereas macros are preprocessed or in simple words, they’re known before execution.

For example, A code in C using macros:- number is the macro here

#include<stdio.h>

#define number 7

int main() {

int multi,prod;

printf(“What do you want 7 to be multiplied with? ”);

scanf(“%d”, &multi);

prod = number\*multi; /\* using macro here \*/

printf(“\n Product = %d”, prod);

}

**Q15. What is “dynamic typing” mechanism? [3 points]**

A15.

Dynamic typing is when in a programming language the data type of the variable is not required to be declared during variable declaration and it’s type checking is performed on run-time.

Python and Lisp use dynamic typing mechanism.

**Extra credit (bonus question):**

**Q16. Give an example of the following (do not use the examples from the slides or the book) : [15 points]**

**a. Inheritance**

**b. Polymorphism**

**c. Abstract Data Type**

A16.

B. Polymorphism

In this method the function whoAmI exists for all smartphone types but at runtime it’s output depends on which object it is invoked on.

public class smartphones{

int price;

string name;

public void whoAmI (){

System.out.println (“I am a Smartphone”);

}

}

public class iPhone extends smartphones{

public void whoAmI (){

System.out.println (“I am an iPhone”);

}

}

public class Android extends smartphones{

public void whoAmI (){

System.out.println (“I am an Android”);

}

}

public class MyBelongings{

public static void main(String[] args){

smartphones a = new smartphones();

smartphones XPro = new iPhone();

smartphones Note10 = new Android();

a.whoAmI();

XPro.whoAmI();

Note10.whoAmI();

}

}