|  |
| --- |
| **External Project Report on**  **Digital Logic Design(EET1211)** |

**N-Bit Adder and Subtractor**



**Submitted by**

|  |  |  |
| --- | --- | --- |
| **A S V K VINAYAK** | **Reg. No.:** | **1941012434** |
| **A S V K VINAYAK** | **Reg. No.:** | **1941012434** |
| **SAMBIT SOURAV PAL** | **Reg. No.:** | **1941012309** |
| **GITESH PANDEY** | **Reg. No.:** | **1941012459** |
| **NIKHIL SAHU**  **JYOTIADITYA PRASAD** | **Reg. No.:**  **Reg. No.:** | **1941012581**  **1941012458** |

**B. Tech. CSE 3rd Semester (Section – P)**

****

# Declaration

We, the undersigned students of B. Tech. of **CSE** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**N-BIT ADDER and SUBTRACTOR**” submitted to **Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar** for the partial fulfillment of the subject **Digital** **Logic Design (EET 1211)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

|  |  |
| --- | --- |
| **A S V K VINAYAK**  **Registration No.: 1941012434** | **SAMBIT SOURAV PAL**  **Registration No.:1941012309** |
| **GITESH PANDEY**  **Registration No.: 1941012459** | **NIKHIL SAHU**  **Registration No.:1941012581** |

**JYOTIADITYA PRASAD**

**Registration No.:** **1941012458**

**DATE: 10/02/2021**

**PLACE: ITER**

# Abstract

For the External Project Report on Digital Logic Design (EE1211) we designed a N- Bit Adder and Subtractor. We used 1 half adder and N-1 full adder for designing N- bit adder, similarly we used 1 half subtractor and N-1 full subtractor for designing N- bit subtractor.

We generated the solution using Truth table. This project was programmed in Xilux software using System Verilog HDL, with the aid of Icarus Verilog 0.9.7 simulation tool. The project involved the design of N- Bit Adder and Subtractor. We also made the block diagram, logic diagram using logic gates.

**Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Serial No.** | **Chapter No.** | | **Title of the Chapter** | **Page No.** |
|  | **1** | | **Introduction** | **01** |
|  | **2** | | **Problem Statement** | **02** |
|  | **3** | | **Methodology** | **03-04** |
|  | **4** | | **Implementation** | **05-07** |
|  | **5** | | **Results and interpretation** | **08** |
|  | **6** | | **Conclusion** | **09** |
|  |  | | **References** | **10** |
|  | |

# **1. Introduction**

The main objective of project is to design and verify different bit adder and subtractor. We have designed a 2 bit adder and subtractor which accepts two 2 bits numbers and the code adder gives the output as the addition of two numbers and subtractor gives the output as the subtraction of two numbers.

For 2 bit addition we have used 1 half adder and 1 full adder, similarly for 2 bit subtraction we have used 1 half subtractor and 1 full subtractor.

In digital electronics adder is a digital circuit that performs addition of numbers, these can be classified into 1 bit adders and multi bit adders. Further 1bit adders are categorized in half adders and full adders. These are not only used in ALU but also in memory for calculating addresses, table indices and many more.

In digital electronics subtractor is a digital circuit that performs subtraction of numbers, these can be classified into 1 bit subtractor and multi bit subtractor. Further 1bit subtractor is categorized in half subtractor and full subtractor. These are not only used in ALU but also in memory for calculating addresses, table indices and many more.

The coding was written in VHDL. The waveforms were obtained successfully. After the coding was done, the synthesis of the code was performed using Xilinx. Synthesis was done; synthesis translates VHDL code into netlist (a textual description). Thereafter, the simulation was done to verify the synthesized code.

# **2. Problem Statement**.

***To Design N bit Adder and Subtractor***

**N-Bit Adder**

The N-bit Adder is simply implemented by connecting 1 Half Adder and N-1 Full Adder in series. The Verilog code for N-bit Adder is designed so that the N value can be initialized independently for each instantiation. To do it, the Verilog code for N-bit Adder uses Generate Statement in Verilog to create a chain of full adders for implementing the N-bit Adder**.**

**N-Bit SUBTRACTOR**

The N-bit Subtractor is simply implemented by connecting 1 Half Subtractor and N-1 Full Subtractor in series. The Verilog code for N-bit Subtractor is designed so that the N value can be initialized independently for each instantiation. To do it, the Verilog code for N-bit Subtractor uses Generate Statement in Verilog to create a chain of full subtractor for implementing the N-bit Subtractor.

**Full Adder:** Full Adder is the adder which adds three inputs and produces two outputs.

**Half Adder:** It**’**s a combinational logic circuit which is designed by connecting one EX-OR gate and one AND gate.

**Full Subtractor:**  It’s a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit.

**Half Subtractor:** A **half subtractor** is a logical circuit that performs a subtraction operation on two binary digits. The **half subtractor** produces a difference and a borrow bit for the next stage.

# **3. Methedology**

1. ***Generating the solution to the problem by the use of Truth table/excitation table, K- map and (or) Boolean algebra.***

**Example:-** We have taken N=2

**2-bit Adder Truth Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A1 | A0 | B1 | B0 | c | S1 | S0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A1 | A0 | B1 | B0 | b | d1 | d0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

# **2-bit Subtractor Truth Table**

1. ***Finding out the different digital ICs to be used in the optimized design*.**

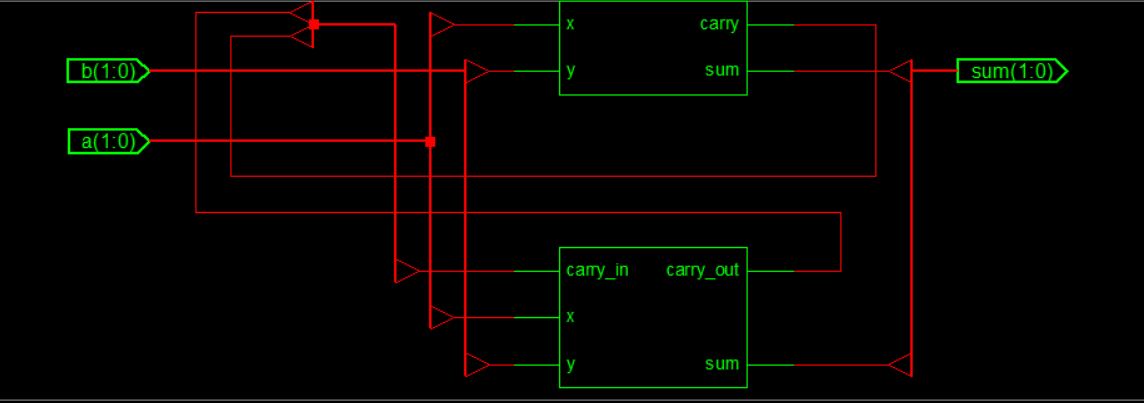
# **ICs**

* **Two input AND gate: 7408 IC**
* **Three input AND gate : 7411 IC**
* **Four input AND gate: 74HC21 IC**
* **Two input XOR gate: CD4070**
* **Two input OR gate: 74HC02**
* **Four input OR gate: 74LS32**
* **Three input OR gate: 74HCT4075**

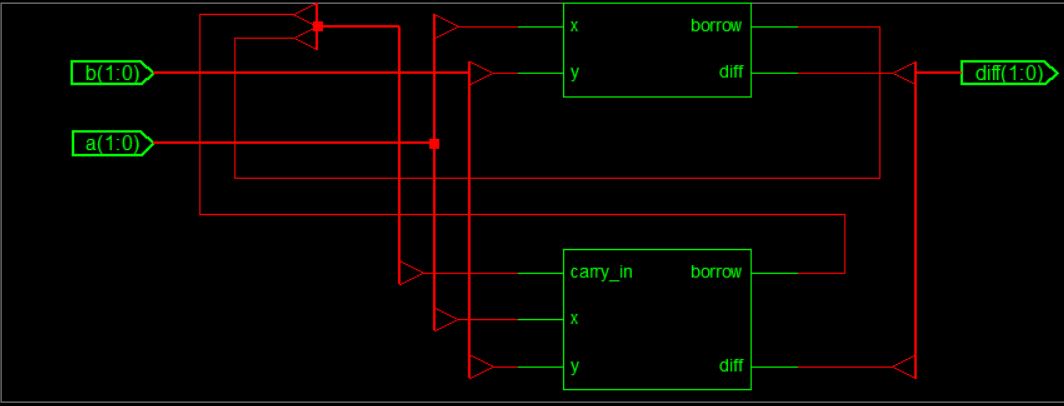
# **4. Implementation**

1. ***Drawing the logic diagram using different logic gates***.

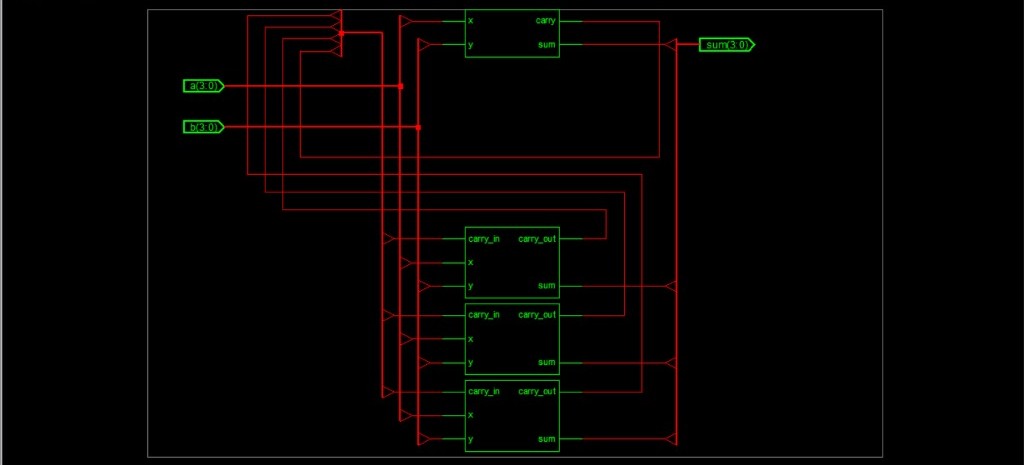
**2-bit Adder logic diagram**



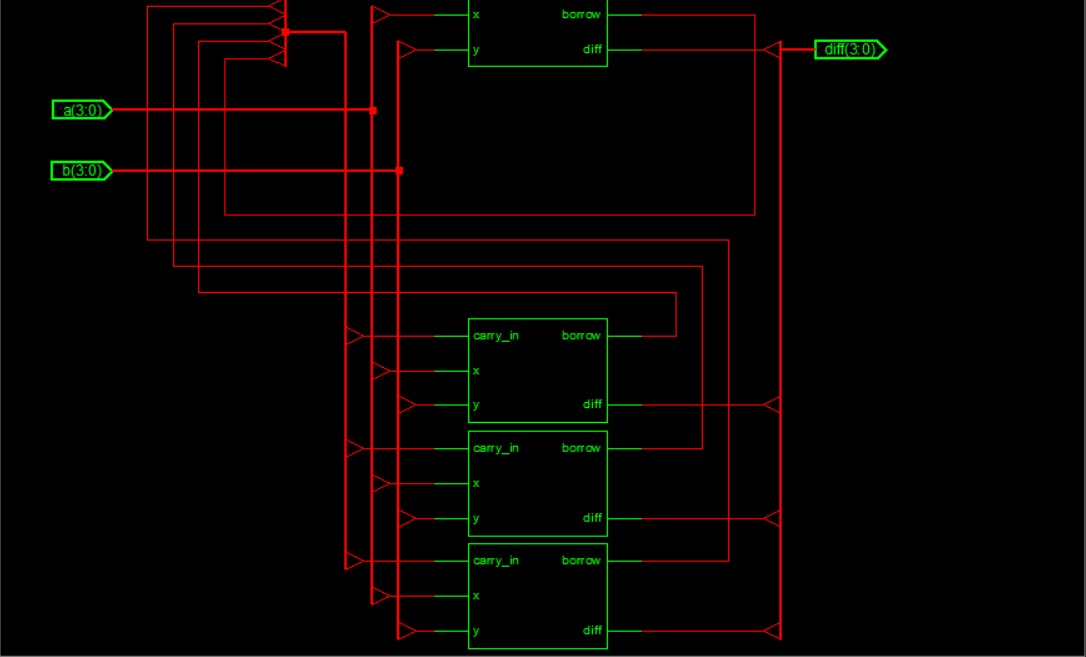
**2-bit Subtractor logic diagram**



**4-bit Adder logic diagram**

****

**4-bit Subtractor logic diagram**

****

**Program:-**

* **FOR ADDER**

|  |  |
| --- | --- |
| **Half Adder** | **Full Adder** |
| module half\_adder(x,y,sum,carry); input x,y;  output sum,carry;  assign sum=x^y;  assign carry=x&&y;  endmodule | module full\_adder(x,y,carry\_in,sum,carry\_out);  input x,y,carry\_in;  output sum,carry\_out;  assign sum = (x^y) ^ carry\_in;  assign carry\_out = (y&&carry\_in)|| (x&&y)  || (x&&carry\_in);  endmodule |

|  |
| --- |
| **N bit Adder**  module N\_bit\_adder(a,b,sum,carry\_out);  parameter N=2;  input [N-1:0] a,b;  output [N-1:0] sum;  output carry\_out;  wire [N-1:0] carry;  genvar i;  generate  for(i=0;i<N;i=i+1)  begin: generate\_N\_bit\_Adder  if(i==0)  half\_adder ha(a[0],b[0],sum[0],carry[0]);  else  full\_adder fa(a[i],b[i],carry[i-1],sum[i],carry[i]);  end  assign carry\_out = carry[N-1];  endgenerate  endmodule |

* **FOR SUBTRACTOR**

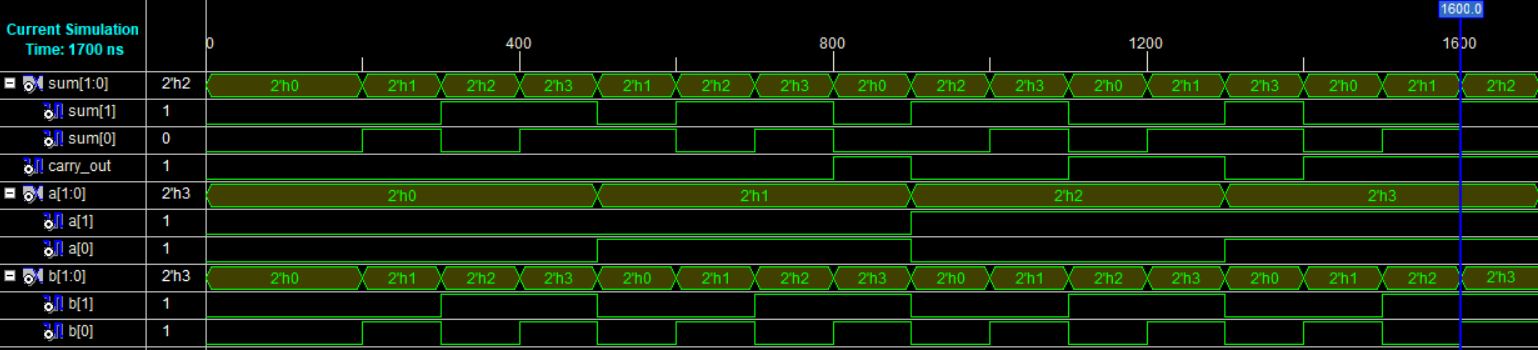
|  |  |
| --- | --- |
| **Half Subtractor** | **Full Subtractor** |
| module half\_substractor( x, y, diff, borrow);  input x, y;  output diff, borrow;  assign diff= x ^ y;  assign borrow= ~x && y;  endmodule | module full\_substractor( x, y, carry\_in, diff, borrow);  input x, y, carry\_in;  output diff, borrow;  assign diff= x ^ y ^ carry\_in;  assign borrow=(( !x)&&(y ^ carry\_in)) || (y && carry\_in);  endmodule |

|  |
| --- |
| **N bit Subtractor**  module N\_bit\_substractor (a,b,diff,borrow\_out); parameter N=2; input [N-1:0] a,b; output [N-1:0] diff; output borrow\_out; wire [N-1:0] br; genvar i; generate  for(i=0;i<N;i=i+1) begin: generate\_N\_bit\_Substractor if(i==0)  half\_substractor hs(a[0],b[0],diff[0],br[0]); else full\_substractor fs(a[i],b[i],br[i-1],diff[i],br[i]); end assign borrow\_out = br[N-1]; endgenerate assign borrow\_out = br[N-1];  endmodule |

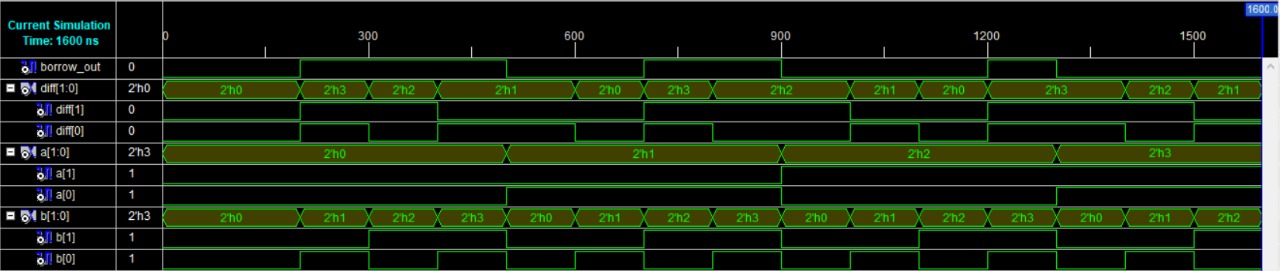
# **5. Results & Interpretation**

Verification of the output for different inputs that satisfies the problem statement by the use of truth table.

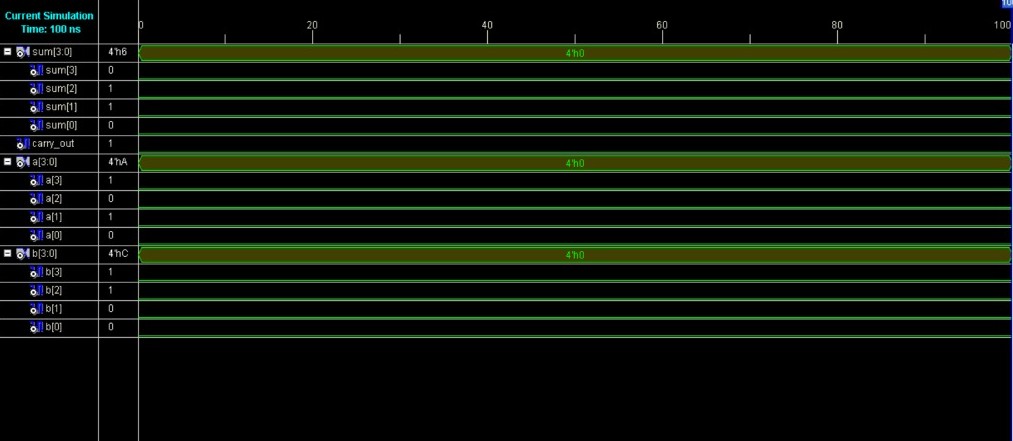
* We can take the value of **N =2**,**4** etc.
* **TBW of 2-bit Adder**:



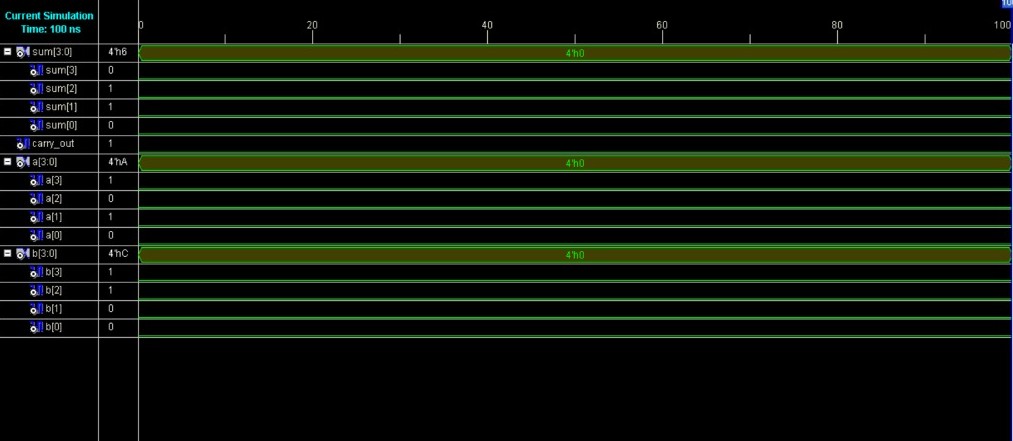
* **TBW of 2-bit Subtractor:**

****

* **TBW of 4-bit Adder**:
* We have taken example of a 4 bit number to perform 4 bit adder.



* **TBW of 4-bit subtractor**
* We have taken example of a 4 bit number to perform 4 bit sbtractor.



**6. Conclusion**

After performing this project on 2-bit adder and subtractor, we got to know many things about various logic gates ICs and designing combinational circuits and writing the HDL program for the problem. In this project we have mainly used full adders and half adders, similarly full subtractor and half subtractor for our objectives. Both adders and subtractors come under the category of combinational logic circuits that are used for arithmetic operations. Adders are used for addition of decimal numbers (inputted in binary form) and subtractors are used for subtracting decimal numbers (inputted in binary form). However, the major difference between half adder and the full adder is that the half adder operates on 2 inputs. On the other hand, full adder operates on 3 inputs. Similarly the major difference between half subtractor and the full subtractor is that the half subtractor operates on 2 inputs. On the other hand, full subtractor operates on 3 inputs.

From this we can conclude that we can add or subtract any bit number from our code.

Our code is so user friendly that user can change the value of n in the verelog code to get the desired bit addition and subtraction.

**REFERENCES**

**(as per the IEEE recommendations)**

# Digital Design: With an Introduction to the Verilog HDL, VHDL, and System Verilog, 6th Edition

* Introduction to Digital Logic Design, Hayes, 1993, Wesley.
* Bhasker, 1997, A Verilog HDL Primer, Allentown
* GeeksForGeeks, tutorialspoint, NESO Academy
* WIKIPIDEA
* Code is on [Github](https://github.com/ASVKVINAYAK/ITER-Minor-projects/tree/master/3RD%20SEMESTER(DL)) anyone can view it.