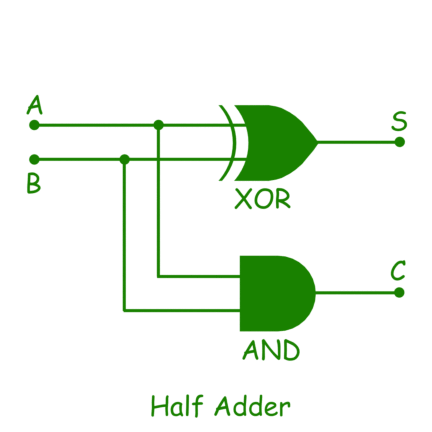
**LAB EXPERIMENT 2**

**Aim:** To design the half adder and full adder in Xilinx software using Verilog level of programming. Design the half adder and full adder using Gate Level of Modelling and Data Flow Modelling.

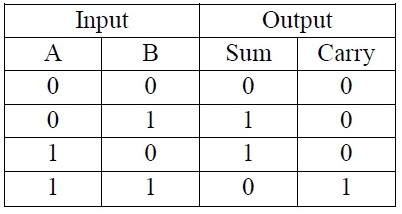
**Theory:**

1. Half Adder: A Half Adder is defined as a basic four terminal digital device which adds two binary input bits. It outputs the sum binary bit and a carry binary bit. As we have defined above, a half adder is a simple digital circuit used to digitally add two binary bits. A binary bit is either 0 or 1

Circuit Diagram:



Truth Table:

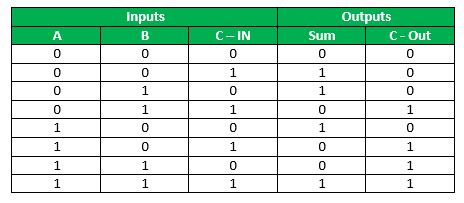


1. Full Adder: Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

Circuit Diagram:



Truth Table:



**Verilog Code and Screenshot of the Program:**

1. **Half Adder by DFM:**
   1. **Verilog Code of the Program**

module DFMhalfadder(

input inp1,

input inp2,

output outs,

output outc

);

assign outs = inp1^inp2;

assign outc = inp1&inp2;

endmodule

* 1. **Screenshot of the Program**

A picture containing text, screenshot, indoor

Description automatically generated

Graphical user interface

Description automatically generated

1. **RTL SCEHMATICS:**

A picture containing text, indoor, screenshot, display

Description automatically generated

A picture containing text, indoor, screenshot

Description automatically generated

1. **Half Adder by GLM:**
2. **Verilog Code of the Program:**

module Abit(

    input a,

    input b,

    output s,

    output c

    );

xor (s,a,b);

and (c,a,b);

endmodule

1. **Screenshot of the Program:**

A picture containing text, screenshot, indoor

Description automatically generated

A picture containing text, screenshot, indoor, computer

Description automatically generated

1. **Full Adder by DFM:**
2. **Verilog Code of the Program**

module FULLADDERDFM(

    input a,

    input b,

    input cin,

    output s,

    output c

    );

wire o1,o2,o3;

assign o1 = a^b;

assign o2 = o1&cin;

assign o3 = a&b;

assign s = o1^cin;

assign c = o2|o3;

endmodule

1. **Screenshots of the Program:**

Graphical user interface, text, application

Description automatically generated

A picture containing text, screenshot, indoor

Description automatically generated

1. **RTL Schematics:**

A picture containing text, indoor, screenshot, display

Description automatically generated

A picture containing text, indoor, screenshot, display

Description automatically generated

1. **Full Adder by GLM:**
2. **Verilog Code of the Program:**

module FulladderSM(

    input x,

    input y,

    input cin,

    output s,

    output c

    );

wire o1,o2,o3;

xor (o1,x,y);

and (o2,x,y);

xor (s,o1,cin);

and (o3,o1,cin);

or (c,o3,o2);

endmodule

1. **Screenshots of the Program:**

Graphical user interface, text, application

Description automatically generated

A picture containing text, screenshot, indoor

Description automatically generated

1. **RTL Schematics:**

A picture containing text, indoor, screenshot, display

Description automatically generated

A picture containing text, indoor, screenshot

Description automatically generated

**Conclusion:** From this experiment we learnt how to design half adder and full adder. We used the gate level modelling technique and data flow modelling technique to design the logic gates. We also have learnt how to check our output on the ISIM Simulator by applying force clock and force value parameters and check our output with the truth table.