**Name: DHVANI PATEL**

**Roll No: 21BCP116**

**Div 2 , Group 4**

AIM: Design a Half Adder, and Full Adder using Half Adder

* **Description:**

# Half Adder (HA):

Half adder is the simplest of all adder circuits. Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (s) and carry bit (c) both as output. The addition of 2 bits is done using a combination circuit called a Half adder. The input variables are augend and addend bits and output variables are sum & carry bits. A and B are the two input bits.

let us consider two input bits A and B, then sum bit (s) is the X-OR of A and B. it is evident from the function of a half adder that it requires one X-OR gate and one AND gate for its construction.

# ha_truthTruth Table:

Table 4.1: Truth table for “Sum” and “Carry”.

Here we perform two operations Sum and Carry, thus we need two K-maps one for each to derive the expression.

* **Logical Expression:**

# For Sum:

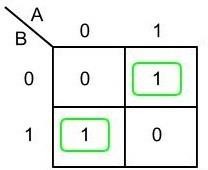


Fig 4.1: K-map for “Sum”

# Sum = A XOR B

# https://media.geeksforgeeks.org/wp-content/uploads/20211017125041/Inkedandkmap1-200x155.jpgFor Carry:

Fig 4.2: K-map for “Carry”

# Carry = A AND B

# halfadder

* **Implementation:**

Fig 4.3: Circuit Diagram of “Half-Adder”.

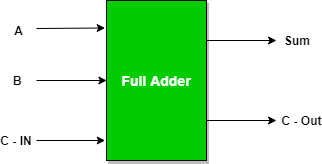
Full Adder is the adder that adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM. A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another. we use a full adder because when a carry-in bit is available, another 1-bit adder must be used since a 1-bit half-adder does not take a carry-in bit. A 1-bit full adder adds three operands and generates 2-bit results.

Fig 4.4: Black box view of “Full Adder”.

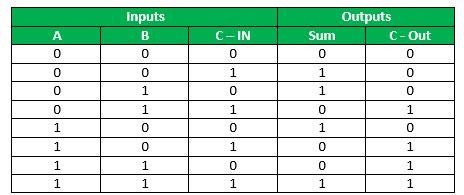


Table 4.2: Truth table of “Full Adder”

**Logical Expression for SUM:** = A’ B’ C-IN + A’ B C-IN’ + A B’ C-IN’ + A B C-IN = C-IN (A’ B’ + A B) + C-IN’ (A’ B + A B’) = C-IN XOR (A XOR B) = (1,2,4,7)

**Logical Expression for C-OUT:** = A’ B C-IN + A B’ C-IN + A B C-IN’ + A B C-IN = A B + B C-IN + A C-IN = (3,5,6,7)

**Another form in which C-OUT can be implemented:** = A B + A C-IN + B C-IN (A + A’) = A B C-IN + A B + A C-IN + A’ B C-IN = A B (1 +C-IN) + A C-IN + A’ B C-IN = A B + A C-IN

+ A’ B C-IN = A B + A C-IN (B + B’) + A’ B C-IN = A B C-IN + A B + A B’ C-IN + A’ B C- IN = A B (C-IN + 1) + A B’ C-IN + A’ B C-IN = A B + A B’ C-IN + A’ B C-IN = AB + C-IN (A’ B + A B’)

Therefore, COUT = AB + C-IN (A EX – OR B)

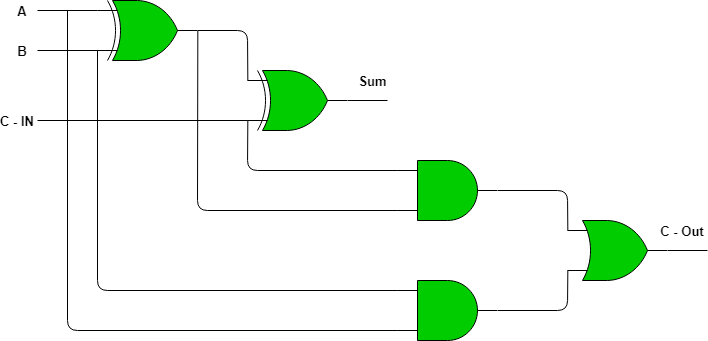


Fig 4.5: Circuit Diagram of “Full Adder”

Implementation of Full Adder using Half Adders:

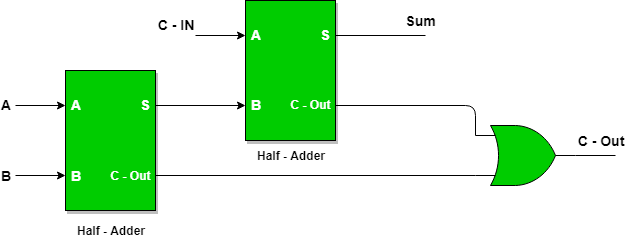
2 Half Adders and an OR gate is required to implement a Full Adder.

Fig 4.6: Circuit Diagram of “Full Adder” using two “Half-Adder”.

With this logic circuit, two bits can be added together, taking a carry from the next lower order of magnitude, and sending a carry to the next higher order of magnitude.

**Question 1. Write Verilog code for half adder. Test using waveform.**

* Code :

// Code your testbench here

module tb\_Q1\_Half\_Adder;

reg A , B ;

wire S , C;

Q1\_Half\_Adder a1(A , B, S, C);

initial

begin

A = 0 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=1; #1;

$display("a= %b , b = %b " , A , B);

$display("x= %b y=%b " , S,C);

A = 1 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=1; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Q1\_Half\_Adder(a,b,x ,y);

input a,b;

output x,y;

wire p , q;

xor( x, a,b);

and( y,a,b);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.

a= 0 , b = 0   
x= 0 y=0   
a= 0 , b = 1   
x= 1 y=0   
a= 1 , b = 0   
x= 1 y=0   
a= 1 , b = 1   
x= 0 y=1   
Finding VCD file...

* Waveform :



**Question 2. Design a four-bit combinational circuit 2’s complement using exclusive-OR gates and half adder. Write Verilog code in Edaplayground and then test using waveform.**

* Code :

// Code your testbench here

module tb\_TwosComplement;

reg A0,A1,A2,A3;

wire B0,B1,B2,B3;

TwosComplement t1(A0,A1,A2,A3,B0,B1,B2,B3);

initial

begin

A0=0;A1=0;A2=0;A3=0;#1;

$display("input= %b %b %b %b " , A0,A1,A2,A3 );

$display("2's = %b %b %b %b" , B0,B1,B2,B3);

A0=0;A1=1;A2=1;A3=0;#1;

$display("input= %b %b %b %b " , A0,A1,A2,A3 );

$display("2's = %b %b %b %b" , B0,B1,B2,B3);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module TwosComplement(A0,A1,A2,A3,B0,B1,B2,B3);

input A0,A1,A2,A3;

output B0,B1,B2,B3;

wire p,q;

or(p,A1,A2,A3);

xor(B0,A0,p);

or(q,A2,A3);

xor(B1,A1,q);

xor(B2,A2,A3);

or(B3,A3,0);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.

input= 0 0 0 0

2's = 0 0 0 0

input= 0 1 1 0

2's = 1 0 1 0

input= 1 1 1 0

2's = 0 0 1 0

Finding VCD file..../dump.vcd

* Waveform :



**Question 3. Write Verilog code for full adder. Test using waveform.**

* Code :

// Code your testbench here

module tb\_Full\_Adder;

reg A , B , Cin ;

wire S , C;

Full\_Adder a1(S,C,Cin, A , B);

initial

begin

Cin=0; A = 0 ; B=0; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=0; A = 0 ; B=1; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B);

$display("x= %b y=%b " , S,C);

Cin=0; A = 1 ; B=0;#1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=0; A = 1 ; B=1; #1;

$display("cin= %b a= %b , b = %b " , Cin,A , B );

$display("x= %b y=%b " , S,C);

Cin=0; A = 0 ; B=0; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=1; A = 0 ; B=1; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B);

$display("x= %b y=%b " , S,C);

Cin=1; A = 1 ; B=0; #1;

$display("cin= %b a= %b , b = %b " ,Cin, A , B );

$display("x= %b y=%b " , S,C);

Cin=1;A = 1 ; B=1; #1;

$display("cin= %b a= %b , b = %b " , Cin,A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Full\_Adder(s,cout,a,b,cin);

input cin,a,b;

output s,cout;

wire p,q ,r, cfull;

xor(p,a,b);

xor(s,p,cin);

and(q,a,b);

and(r,p,cin);

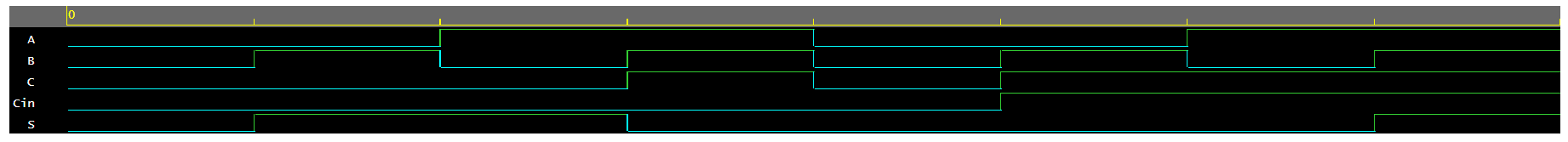
or(cout,q,r);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.  
cin= 0 a= 0 b = 0   
x= 0 y=0   
cin= 0 a= 0 b = 1   
x= 1 y=0   
cin= 0 a= 1 b = 0   
x= 1 y=0   
cin= 0 a= 1 b = 1   
x= 0 y=1   
cin= 0 a= 0 b = 0   
x= 0 y=0   
cin= 1 a= 0 b = 1   
x= 0 y=1   
cin= 1 a= 1 b = 0   
x= 0 y=1   
cin= 1 a= 1 b = 1   
x= 1 y=1

* Waveform :



**Question 4. Write Verilog code for full adder using module instantiation of half adder. Test using waveform.**

* Code :

module tb\_Full\_Adder;

reg A , B,CIN ;

wire S , C;

Full\_Adder a1(CIN,A , B, S, C);

initial

begin

A = 0 ; B=0; CIN=0; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=0; CIN=1; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=1;CIN=0; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=1;CIN=1; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=0;CIN=0; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=0;CIN=1; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=1; CIN=0; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=1; CIN=1; #1;

$display("a= %b , b = %b, cin= %b " , A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Q1\_Half\_Adder(a,b,x ,y);

input a,b;

output x,y;

xor( x, a,b);

and( y,a,b);

endmodule

module Full\_Adder(cin ,a,b, s,cout):

input cin,a,b;

output s,cout;

wire p,q , cfull;

Q1\_Half\_Adder a1(a,b, p,q);

Q1\_Half\_Adder a2(cin,p, s, cfull);

or(cout , cfull, q);

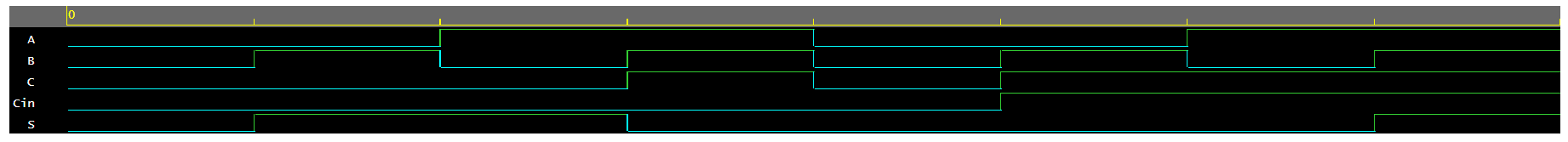
endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.

cin= 0 a= 0 b = 0   
x= 0 y=0   
cin= 0 a= 0 b = 1   
x= 1 y=0   
cin= 0 a= 1 b = 0   
x= 1 y=0   
cin= 0 a= 1 b = 1   
x= 0 y=1   
cin= 0 a= 0 b = 0   
x= 0 y=0   
cin= 1 a= 0 b = 1   
x= 0 y=1   
cin= 1 a= 1 b = 0   
x= 0 y=1   
cin= 1 a= 1 b = 1   
x= 1 y=1

* Waveform :



**Question 5. Write Verilog code for half adder using only NAND gate. Test using waveform.**

* Code :

// Code your testbench here

module tb\_Q1\_Half\_Adder;

reg A , B ;

wire S , C;

Q1\_Half\_Adder a1(A , B, S, C);

initial

begin

A = 0 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=1; #1;

$display("a= %b , b = %b " , A , B);

$display("x= %b y=%b " , S,C);

A = 1 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=1; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Q1\_Half\_Adder(a,b,x ,y);

input a,b;

output x,y;

wire p , q;

nand(a\_not ,a);

nand(b\_not , b);

nand(p , a,b);

nand(q , p,a);

nand(r , p,b);

nand(y,q,r);

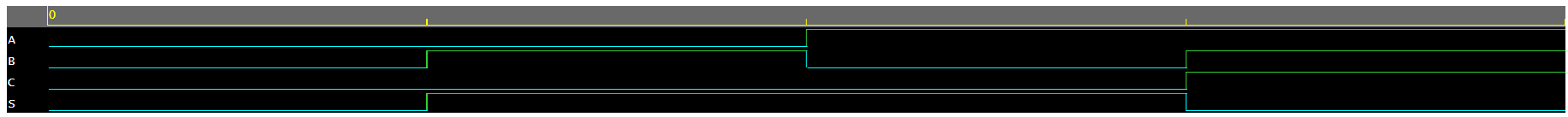
nand(y,p);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.  
a= 0 , b = 0   
x= 0 y=0   
a= 0 , b = 1   
x= 1 y=0   
a= 1 , b = 0   
x= 1 y=0   
a= 1 , b = 1   
x= 0 y=1   
Finding VCD file...

* Waveform :



**Question 6. Write Verilog code for half adder using only NOR gate. Test using waveform.**

* Code :

// Code your testbench here

module tb\_Q1\_Half\_Adder;

reg A , B ;

wire S , C;

Q1\_Half\_Adder a1(A , B, S, C);

initial

begin

A = 0 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 0 ; B=1; #1;

$display("a= %b , b = %b " , A , B);

$display("x= %b y=%b " , S,C);

A = 1 ; B=0; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

A = 1 ; B=1; #1;

$display("a= %b , b = %b " , A , B );

$display("x= %b y=%b " , S,C);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

// Code your design here

module Q1\_Half\_Adder(a,b,x ,y);

input a,b;

output x;

wire p , q,y;

nor(a\_not ,a);

nor(b\_not , b);

nor(p , a,b);

nor(y, a\_not , b\_not);

nor( x,p,y);

endmodule

* Output :

VCD info: dumpfile dump.vcd opened for output.  
a= 0 , b = 0   
x= 0 y=0   
a= 0 , b = 1   
x= 1 y=0   
a= 1 , b = 0   
x= 1 y=0   
a= 1 , b = 1   
x= 0 y=1   
Finding VCD file...

* Waveform :

