Digital Systems Design Lab

Course Code: BECE102P

Submitted to -

Dr. Sathya Sree J

Submitted by -

Sidak Singh 24BCE1460

Akshit Madathil 24BCE5304

EXPERIMENT 1:
Verification of Boolean expression/Logic gates:(Hardware)

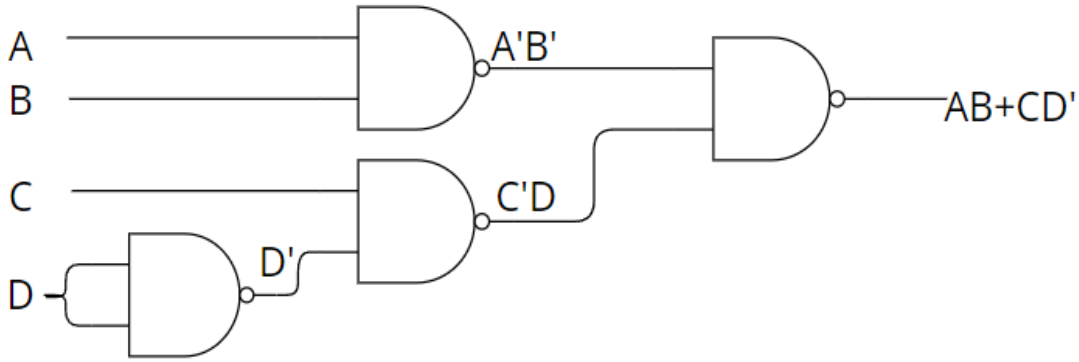
Aim:

To verify the Boolean expression AB+CD’ .

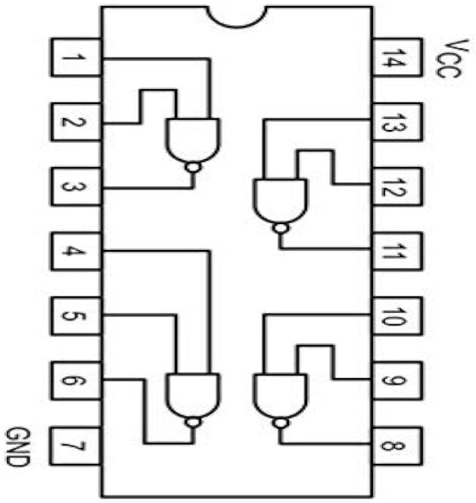
Components Required:

S. No.	COMPONENT	SPECIFICATION	QUANTITY
1.	2-INPUT NAND GATE	IC 7400	1
2.	DIGITAL TRAINER KIT	-	1
3.	Connecting wires	-	few

Logic Diagram:



Pin Diagram:



Truth Table:

Theoretical:-

INPUT				OUTPUT
A	**B**	**C**	**D**	**AB+CD’**
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Experimental:-

A	**B**	**C**	**D**	**AB+CD’**
0	0	0	0	0

0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Experiment 1: Verification of Boolean expression/Logic gates

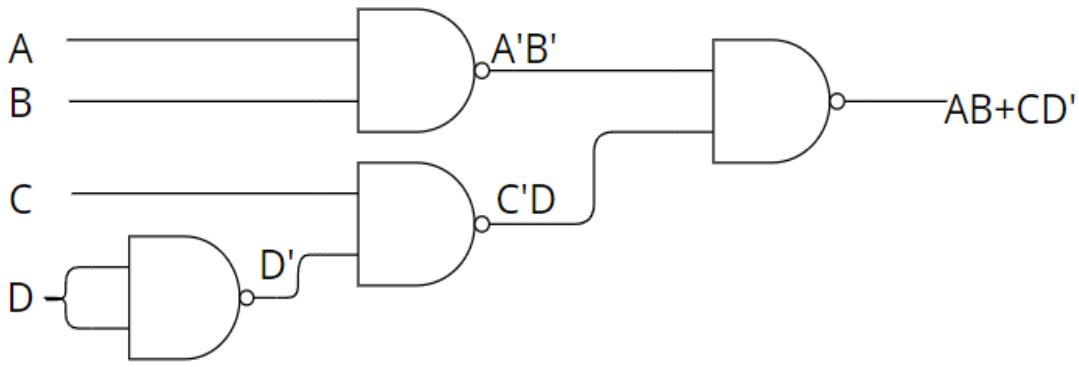
Aim:

To verify the Boolean expression AB+CD’ .

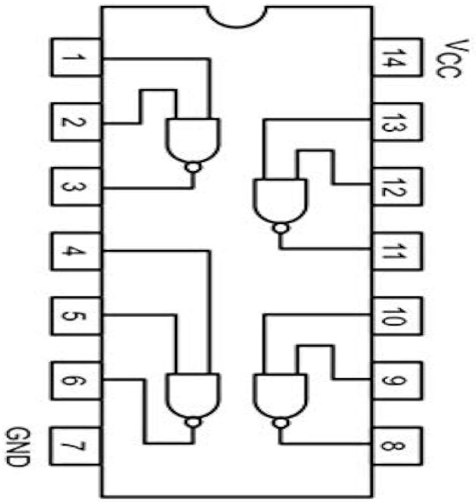
Components Required:

S. No.	COMPONENT	SPECIFICATION	QUANTITY
1.	2-INPUT NAND GATE	IC 7400	1
2.	DIGITAL TRAINER KIT	-	1
3.	Connecting wires	-	few

Logic Diagram:



Pin Diagram:



Truth Table:

Theoretical:-

INPUT				OUTPUT
A	**B**	**C**	**D**	**AB+CD’**
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Experimental:-

A	**B**	**C**	**D**	**AB+CD’**
0	0	0	0	0

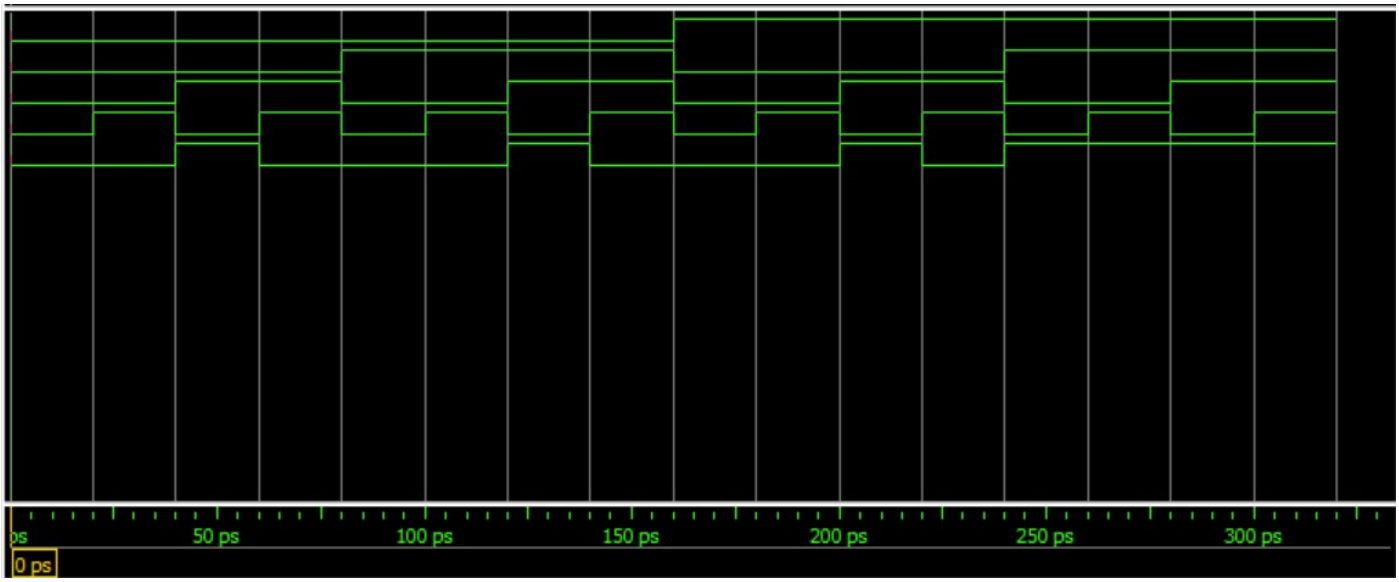
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Code:

```
// Code for expresion AB+CD' using NAND gates.
module practice(a,b,c,d,y);
input a;
input b;
input c;
input d;
output y;
wire t1;
wire t2;
wire t3;
nand(t1,a,b);
nand(t2,d,d);
nand(t3,t2,c);
nand(y,t3,t1);
endmodule
```

```
// Test bench for expression AB+CD' using NAND gates.
module testp;
reg a;
reg b;
reg c;
reg d;
wire y;
practice uut (a,b,c,d,y);
initial
begin
a = 0; b = 0;c = 0;d = 0;
#20 a = 0; b = 0;c = 0;d = 1;
#20 a = 0; b = 0;c = 1;d = 0;
#20 a = 0; b = 0;c = 1;d = 1;
#20 a = 0; b = 1;c = 0;d = 0;
#20 a = 0; b = 1;c = 0;d = 1;
#20 a = 0; b = 1;c = 1;d = 0;
#20 a = 0; b = 1;c = 1;d = 1;
#20 a = 1; b = 0;c = 0;d = 0;
#20 a = 1; b = 0;c = 0;d = 1;
#20 a = 1; b = 0;c = 1;d = 0;
#20 a = 1; b = 0;c = 1;d = 1;
#20 a = 1; b = 1;c = 0;d = 0;
#20 a = 1; b = 1;c = 0;d = 1;
#20 a = 1; b = 1;c = 1;d = 0;
#20 a = 1; b = 1;c = 1;d = 1;
#20 $stop;
$monitor($time, "a=%b,b=%b,c=%b,d=%b,y=%b", a,b,c,d,y);
$dumpfile("dump.vcd");
$dumpvars();
end
endmodule
```

Graph:



Result:

The output of the boolean expression AB+CD’ verified using boolean algebra and logic gate simulations.

Aim: -

Implementation of Half adder using logic gates.

APPARATUS REQUIRED

Components Required:

S. No.	COMPONENTS	SPECIFICATION	QUANTITY
1.	2-INPUT AND GATE	IC 7408	1
2.	2 INPUT XOR GATE	IC 7486	1
3.	DIGITAL TRAINER KIT	-	1
4.	Connecting wires	-	few

THEORY:

Half-Adder: A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bit, S, and the other is the carry bit, C. The Boolean functions describing the half-adder are:

$S = A \oplus B$

$C = A\ B$

Procedure: -

1. Verify the gates.

2. Make the connections as per the circuit diagram.

3. Switch on VCC and apply various combinations of input according to the

truth table.

4. Note down the output readings for half and full adder sum and the carry

bit for different combinations of inputs.

Truth Table:

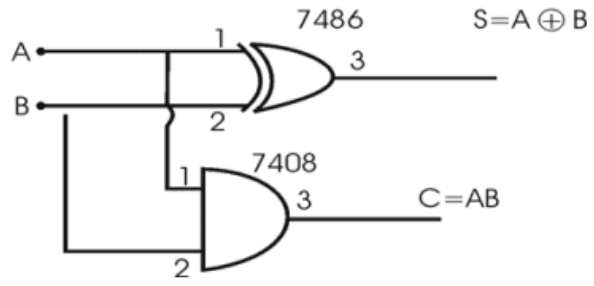
Theoretical:-

INPUT		SUM	CARRY
A	B	A ⊕ B	AB
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Experimental:-

INPUT		SUM	CARRY
A	B	A ⊕ B	AB
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder using basic gates: -



OBSERVATION :

As per truth table we verified the output of Half –Adder

RESULT:

We have designed Half adder & verified their outputs with truth table.

Experiment 2:

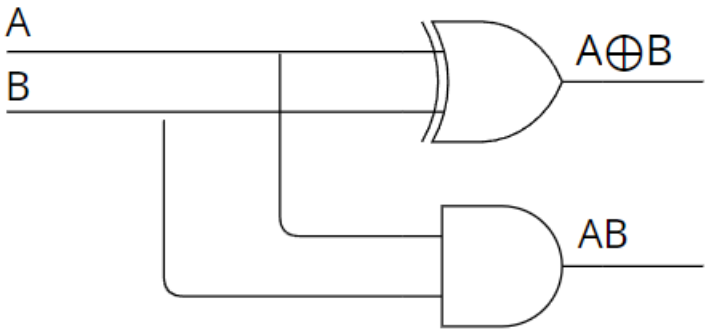
Verification of Half Adder using data flow method.

Aim: To verify truth table of half adder using data flow method.

Components Required:

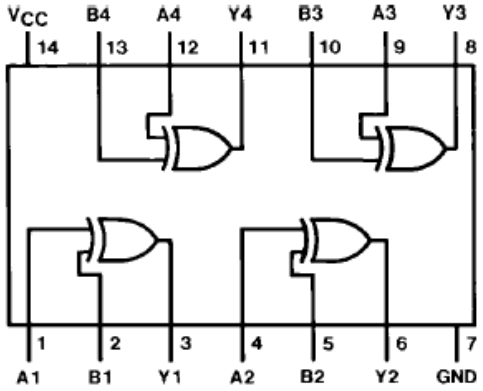
S. No.	COMPONENTS	SPECIFICATION	QUANTITY
1.	2-INPUT AND GATE	IC 7408	1
2.	2 INPUT XOR GATE	IC 7486	1
3.	DIGITAL TRAINER KIT	-	1
4.	Connecting wires	-	few

Logic Diagram:



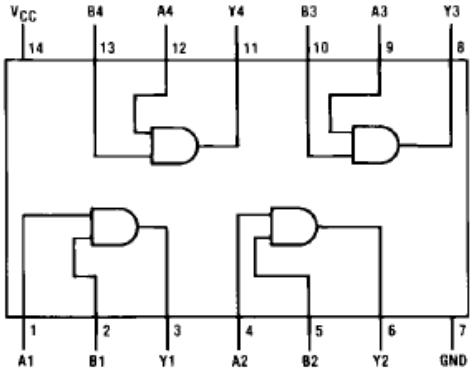
Pin Diagram:

7486



Exclusive-OR Gate
(xor)

7408



AND Gate

Truth Table:

Theoretical:-

INPUT		SUM	CARRY
A	B	A⊕B	AB
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Experimental:-

INPUT		SUM	CARRY
A	B	A⊕B	AB
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

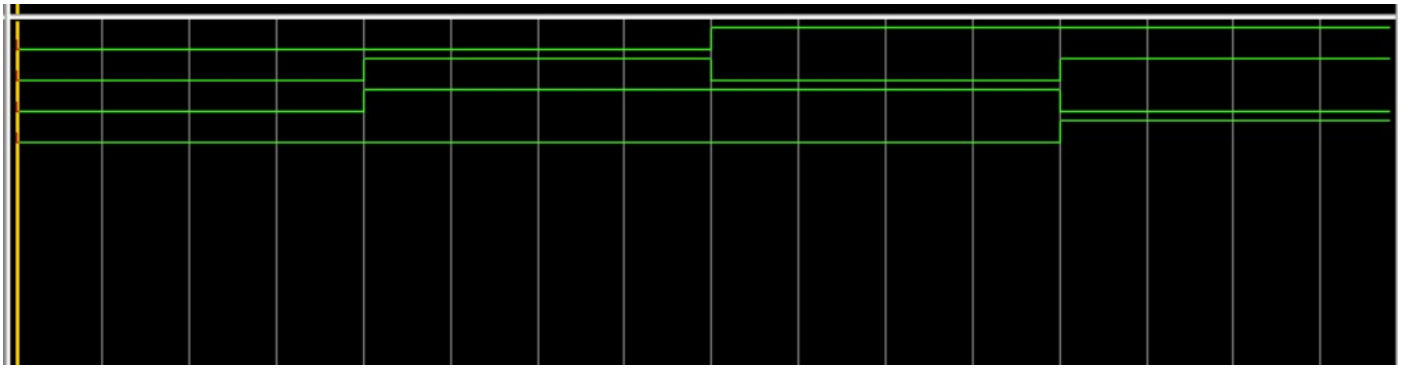
Code:

// Code for half adder using data flow modelling.

```
module practice(a,b,s,c);
input a;
input b;
output s;
output c;
assign s=a^b;
assign c=a&b;
endmodule

// Test bench for Half Adder.
module testp;
reg a;
reg b;
wire s;
wire c;
practice uut (a,b,s,c);
initial
begin
a = 0; b = 0;
#20 a = 0; b = 1;
#20 a = 1; b = 0;
#20 a = 1; b = 1;
#20 $stop;
$monitor($time, "a=%b,b=%b,s=%b,c=%b", a,b,s,c);
$dumpfile("dump.vcd");
$dumpvars();
end
endmodule
```

Graph:



Result:

The truth table for half adder is verified in both hardware and simulation using data flow method.

EXPERIMENT

Aim: -

Implementation of Full adder using logic gates

Apparatus required:-

Component	Description	IC Number
AND Gate	For computing A & B (AND operation)	7408 (Quad 2-input AND Gate)
OR Gate	For computing the sum (OR operation)	7432 (Quad 2-input OR Gate)
XOR Gate	For computing A ⊕ B (XOR operation)	7486 (Quad 2-input XOR Gate)
NOT Gate (Inverter)	For inverting signals (needed for the carry-in)	7404 (Hex Inverter)
Full Adder IC (Optional)	Prebuilt full adder IC for simplicity	7483 (4-bit Full Adder)

THEORY:

Full-Adder: A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin, is called a full-adder. The Boolean functions describing the full-adder are:

$S = (x \oplus y) \oplus Cin$

$C = xy + Cin\ (x \oplus y)$

Procedure: -

1. Verify the gates.

2. Make the connections as per the circuit diagram.

3. Switch on VCC and apply various combinations of input according to the

truth table.

4. Note down the output readings for full adder sum and the carry

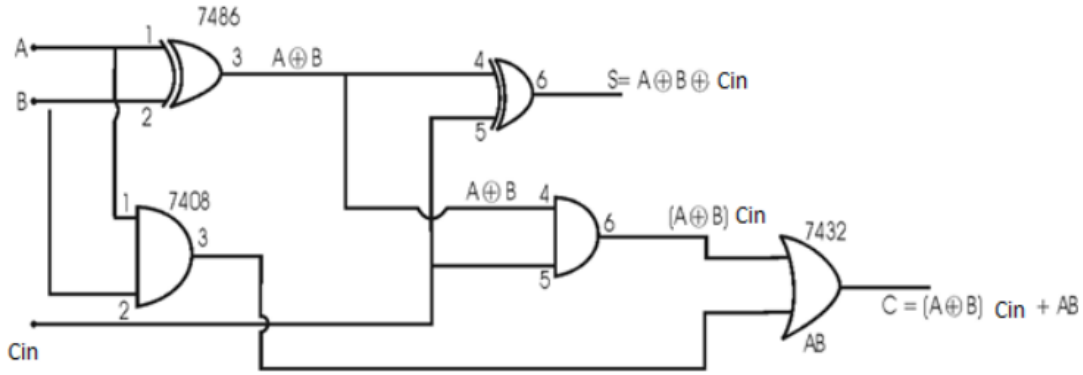
bit for different combinations of inputs.

Full Adder Truth table

INPUTS			OUTPUTS	
A	B	C_{in}	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder using basic gates: -

Full Adder using basic gates:-



OBSERVATION :

As per truth table we verified the output of Full–Adder

RESULT:

We have designed Full adder & verified their outputs with truth table.

Experiment 2:

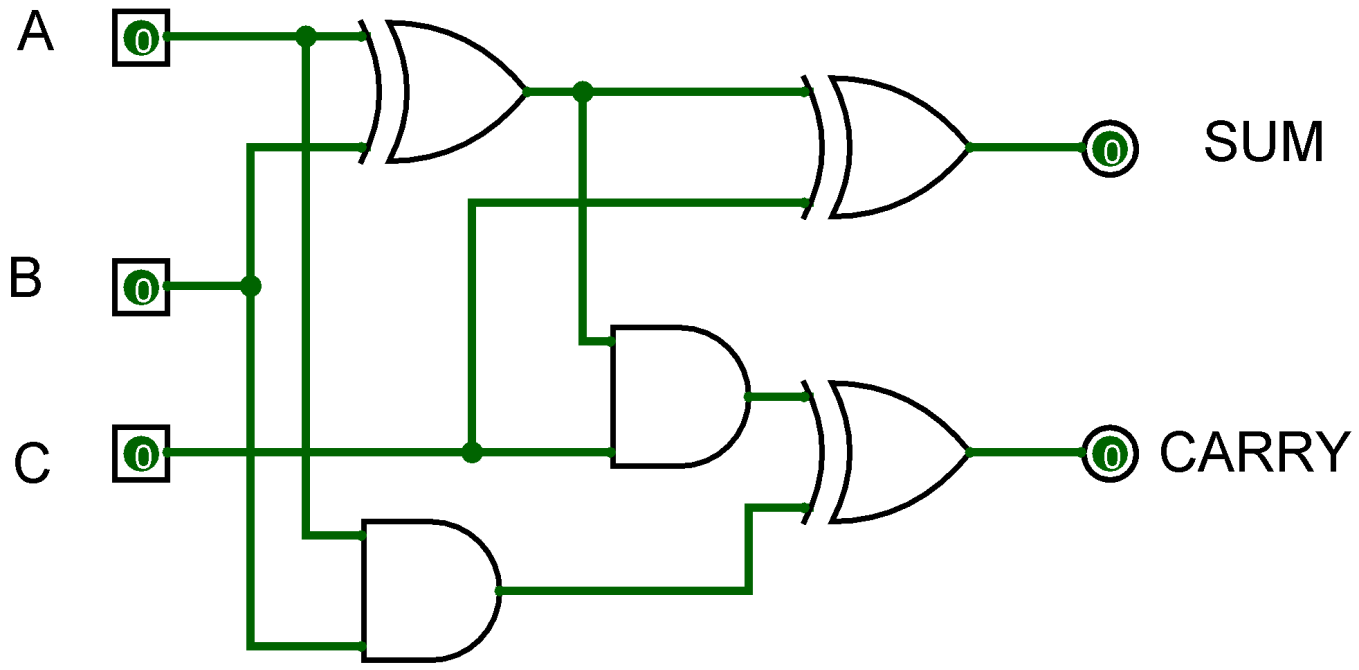
Verification of Full Adder using data flow method.

Aim: To verify truth table of full adder using data flow method.

Components Required:

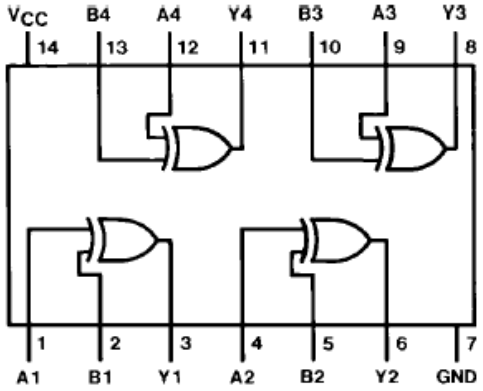
S. No.	COMPONENTS	SPECIFICATION	QUANTITY
1.	INPUT AND GATE	IC 7408	2
2.	INPUT XOR GATE	IC 7486	3
3.	DIGITAL TRAINER KIT	-	1
4.	Connecting wires	-	few

Logic Diagram:



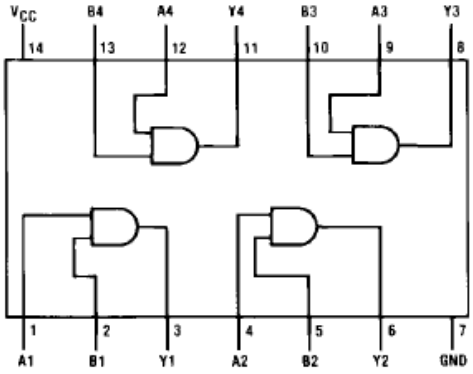
Pin Diagram:

7486



Exclusive-OR Gate
(xor)

7408



AND Gate

Truth Table:

Theoretical:-

Input			Output	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Experimental:-

Input			Output	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Code:

Code for full adder using data flow modelling.

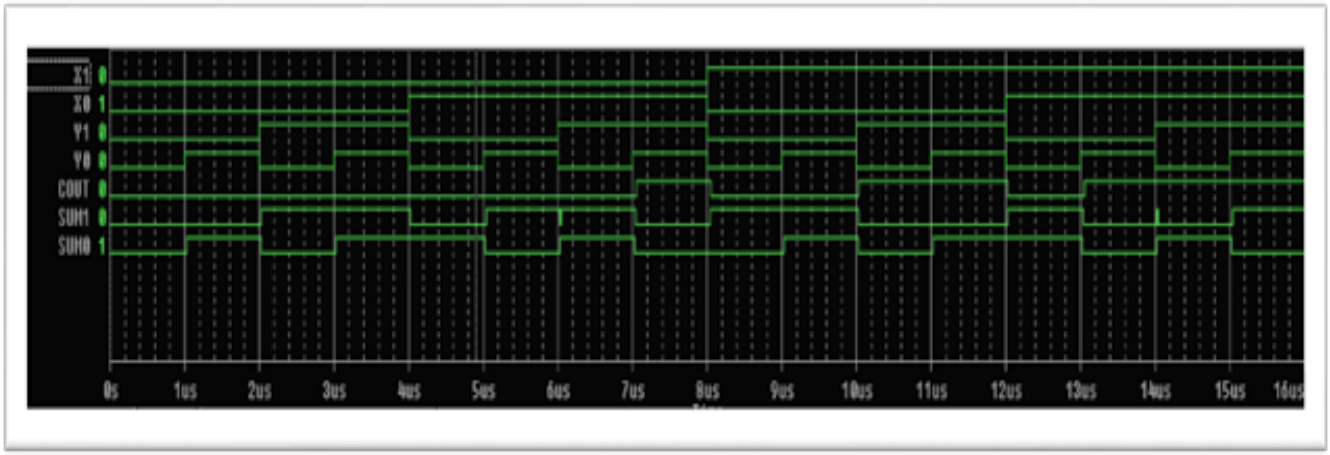
```
module full_adder_d (
   input a,b,cin,
   output sum,carry
);
assign sum = a ^ b ^ cin;
assign carry = (a & b) | (b & cin)  | (cin & a) ;
endmodule
```

Test bench for Full Adder.

```
module full_adder_tb;
reg a,b,cin;
wire sum,carry;
full_adder_s uut(a,b,cin,sum,carry);
initial begin
a = 0; b = 0; cin = 0;
#10
a = 0; b = 0; cin = 1;
#10
a = 0; b = 1; cin = 0;
#10
```

```
a = 0; b = 1; cin = 1;
#10
a = 1; b = 0; cin = 0;
#10
a = 1; b = 0; cin = 1;
#10
a = 1; b = 1; cin = 0;
#10
a = 1; b = 1; cin = 1;
#10
$finish();
end
endmodule
```

Graph:



Result:

The truth table for full adder is verified in both hardware and simulation using data flow method.

AIM: -

Implementation of half subtractor using logic gates.

APPARATUS REQUIRED

1.IC 7486, IC 7432, IC 7408, IC 7404, IC 7400.

2. BreadBoard.

THEORY:

HALF SUBTRACTOR: Subtracting a single-bit binary value B from another A (i.e. A -B) produces a difference bit D and a borrow outbit B-out. This operation is called half subtraction and the circuittorealize it is called a half subtractor. The Boolean functions describing the half Subtractor are:

$D = A \oplus B$ $Br = \bar{A}B$

Procedure: -

1. Verify the gates.

2. Make the connections as per the circuit diagram.

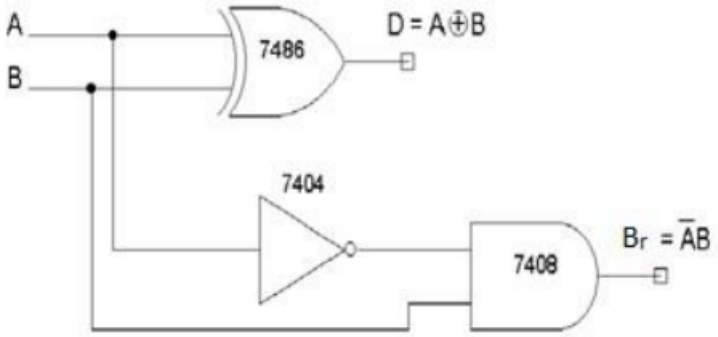
3. Switch on VCC and apply various combinations of input according to thetruth table.

4. Note down the output readings for half subtractor difference and borrow bit for different combinations of inputs.

Truth Table:-

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Using Basic Gates (a)Half Subtractor:-



Conclusion: -

Half subtractor are constructed and their truth tables are Verified.

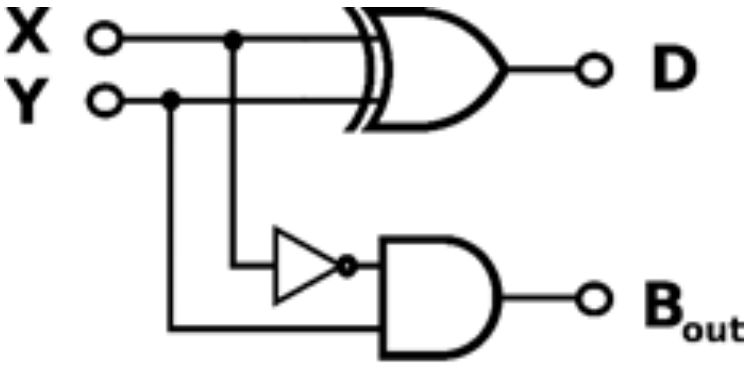
Verification of Half subtractor using data flow method.

Aim: To verify truth table of half Subtractor using data flow method.

Components Required:

S. No.	COMPONENTS	SPECIFICATION	QUANTITY
1.	INPUT AND GATE	IC 7408	2
2.	INPUT XOR GATE	IC 7486	3
3.	INPUT NOT GATE	IC 7404	1
4.	DIGITAL TRAINER KIT	-	1
5.	Connecting wires	-	few

LOGIC DIAGRAM:-



.

Truth Table:-

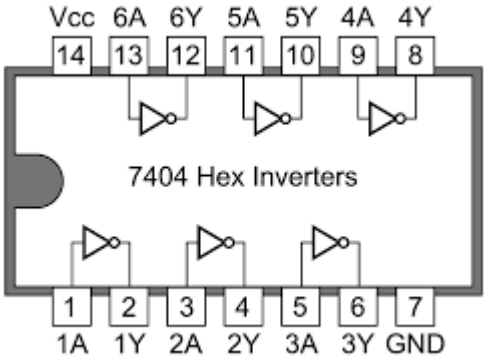
Theoretical

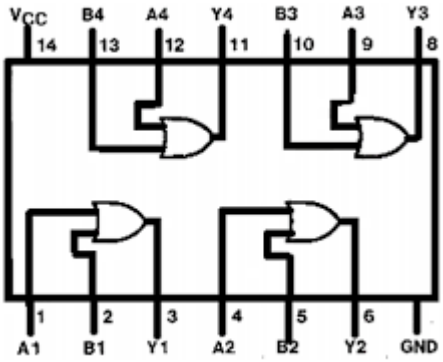
Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

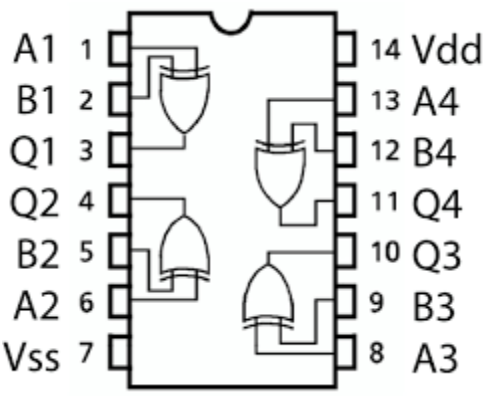
Experimental

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

PIN DIAGRAM







CODE:

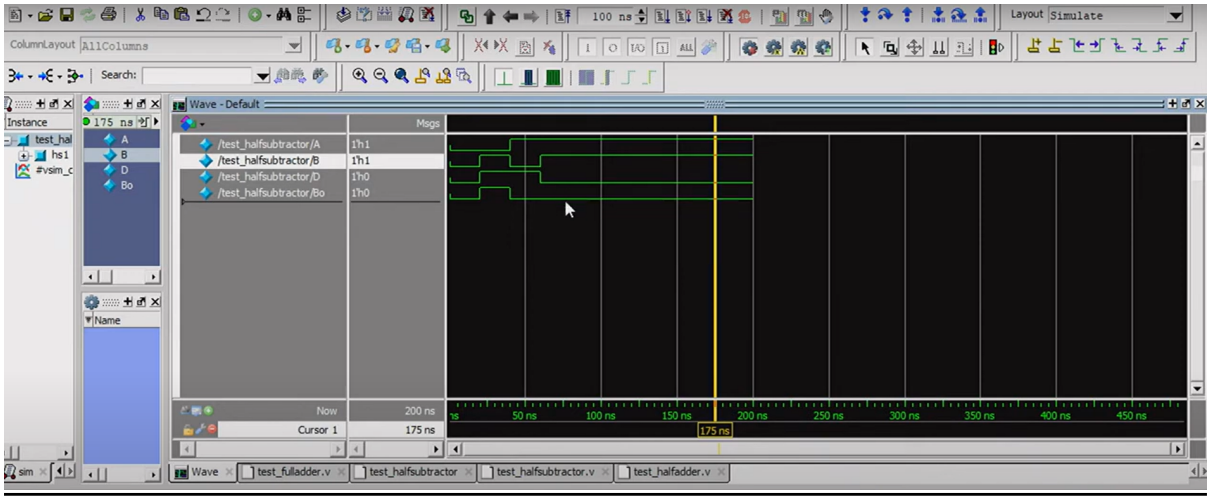
```
module HS(a , b , D, bor);
input a,b;
output D,bor;
assign D=a^b;
assign bor=(~a)&amp;b;
endmodule
```

TEST BENCH

```
module hs_tb;
reg a;
reg b;
```

```
wire D,bor;
HS uut(
.D(D),
.bor(bor),
.a(a),
.b(b)
);
initial
begin
a=0;
b=0;
#2 a=1'b1;b=1'b0;
#2 a=1'b0;b=1'b1;
#2 a=1'b1;b=1'b1;
end
initial $monitor("time=%g. D=%b ,bor=%b, a=%b ,b=%b",$time,D,bor,a,b);
initial #10 $finish;
endmodule
```

Graph:-



Conclusion: -

Half subtractor are constructed and their truth tables are Verified.

AIM: - Implementation of Full subtractor using logic gates.

APPARATUS REQUIRED:-

1.IC 7486, IC 7432, IC 7408, IC 7404, IC 7400.

2. BreadBoard.

THEORY:-

FULL SUBTRACTOR: Subtracting two single-bit binary values, B, Cin from a single-bit value A produces adifference bit D and a borrowout Br bit. This is called full subtraction. The Boolean functionsdescribing the full-subtracter are:

$D = (x \oplus y) \oplus Bin$ $Br = \bar{A}B + \bar{A}(Bin) + B\,(Bin)$

Procedure: -

1. Verify the gates.

2. Make the connections as per the circuit diagram.

3. Switch on VCC and apply various combinations of input

according to thetruth table.

4. Note down the output readings for full

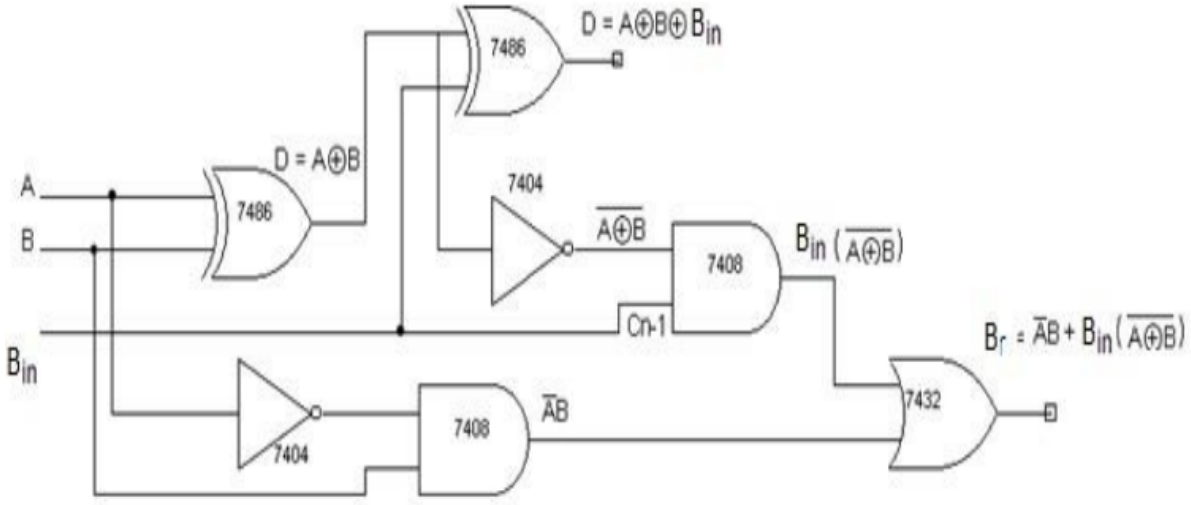
subtractor difference and borrow bit for different

combinations of inputs.

Truth Table:-

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full Subtractor:-



Conclusion: -

full subtractor are constructed and their truth tables are Verified.

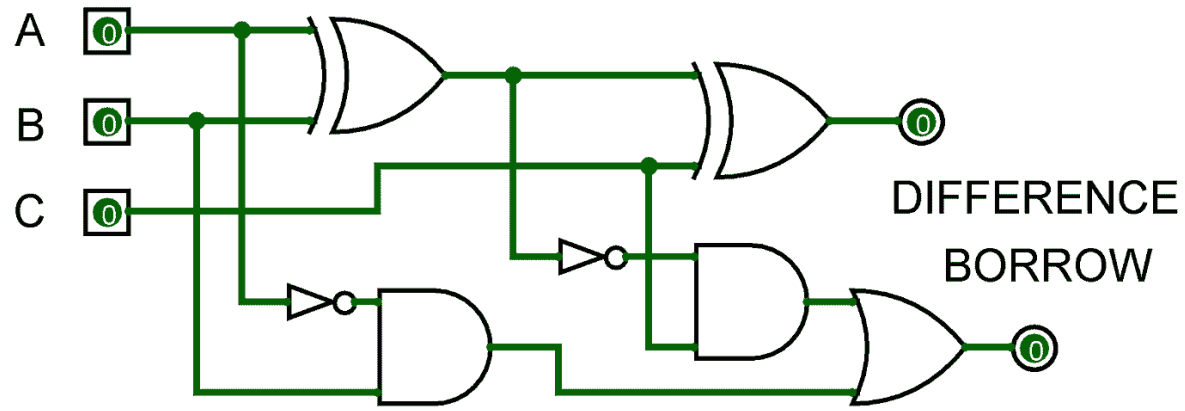
Verification of Full subtractor using data flow method.

Aim: To verify truth table of full Subtractor using data flow method.

Components Required:

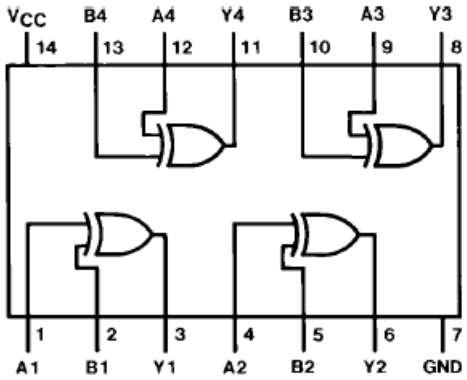
S. No.	COMPONENTS	SPECIFICATION	QUANTITY
1.	INPUT AND GATE	IC 7408	2
2.	INPUT XOR GATE	IC 7486	3
3.	INPUT OR GATE	IC 7432	1
4.	DIGITAL TRAINER KIT	-	1
5.	Connecting wires	-	few

Logic Diagram



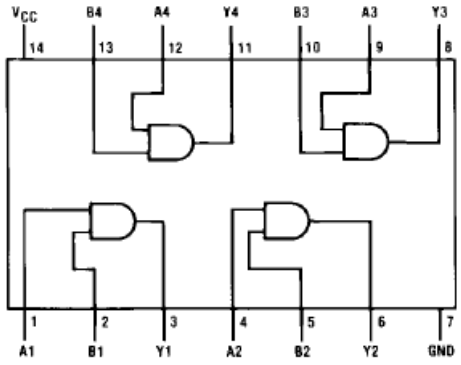
PIN DIAGRAM:-

7486



Exclusive-OR Gate
(xor)

7408



AND Gate

Truth Table:-

Theoritical

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Experimental

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

CODE:

```
module FS(a,b,c,D,bor);
input a,b,c;
output D,bor;
assign D=a^b^c;
assign bor=((~a)&amp;amp;c)|((~a)&amp;amp;b)|(b&amp;amp;c);
endmodule
```

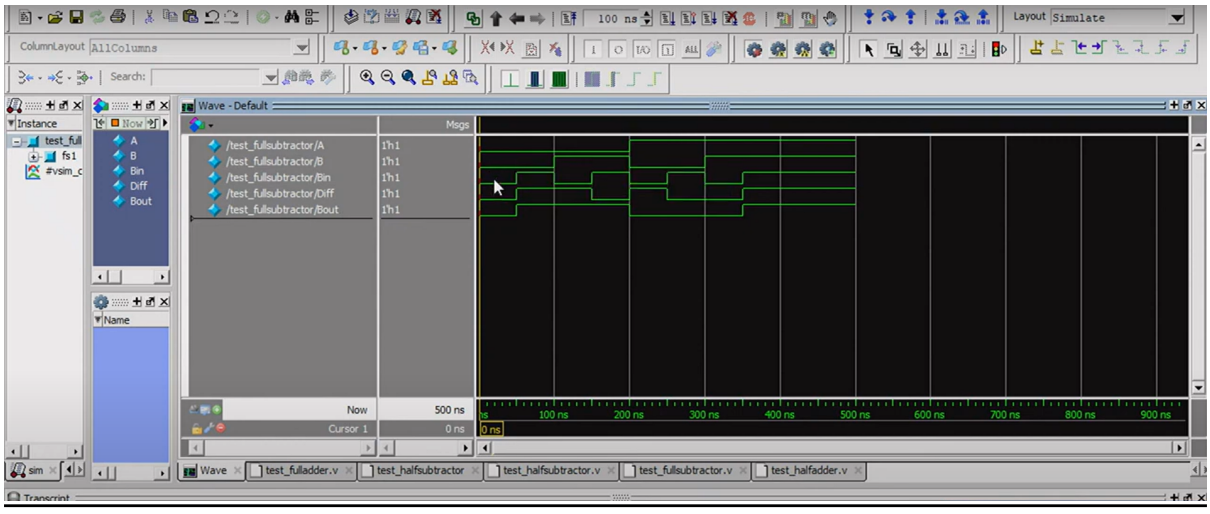
TEST BENCH

```
module FS_tb;
reg a;
reg b;
reg c;
wire D;
wire bor;
FS uut(
.D(D),
.bor(bor),
```

```
.a(a),
.b(b),
.c(c)
);
initial begin
a=0;
b=0;
c=0;
end
always #4 a=a+1'b1;
always #2 b=b+1'b1;
always #1 c=c+1'b1;

initial #20 $finish;
endmodule;
```

Graph:-



Conclusion: -

full subtractor are constructed and their truth tables are Verified.