## Laboratory Report Cover Sheet

**18ECE206J ADVANCED DIGITAL SYSTEMS DESIGN**

**Fourth Semester, 2021-22 (Even semester)**

SRM Institute of Science and Technology College of Engineering and Technology

Department of Electronics and Communication Engineering

#### Name :

**Register No. :**

**Day/ Session :**

**Venue :**

**Title of Experiment :**

**Date of Conduction :**

**Date of Submission :**

|  |  |  |
| --- | --- | --- |
| **Particulars** | **Max. Marks** | **Marks**  **Obtained** |
| Pre lab and Post lab | 10 |  |
| Lab Performance | 20 |  |
| Simulation and results | 10 |  |
| Total | 40 |  |

**REPORT VERIFICATION**

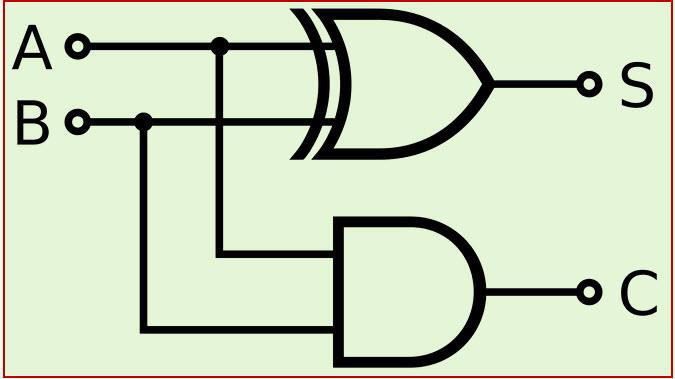
**Staff Name : Signature :**

# 6. **Implementation of Adder and Subtractor Using Data flow, Structural and behavioral modeling**

**Aim:** To design and implement adder and Subtractor Using Data flow, Structural and behavioral modeling

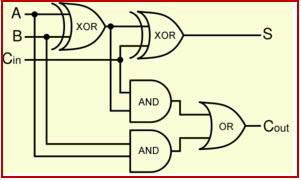
**Software requirement:**  Xilinx ise & Modelsim

**Theory:** An adder is a [digital logic circuit](https://www.elprocus.com/different-types-of-digital-logic-circuits/) in electronics that implements addition of numbers. In many computers and other types of processors, adders are used to calculate addresses, similar operations and table indices in the ALU and also in other parts of the processors. These can be built for many numerical representations like excess-3 or binary coded decimal. Adders are classified into two types: half adder and full adder. The half adder circuit has two inputs: A and B, which add two input digits and generate a carry and sum. The full adder circuit has three inputs: A and C, which add the three input numbers and generate a carry and sum.



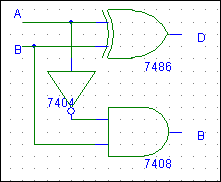
|  |  |  |  |
| --- | --- | --- | --- |
| A | B | Sum | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**Fig 8: Half adder circuit Diagram** Truth table of half adder



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | S | Cout |
| 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 |

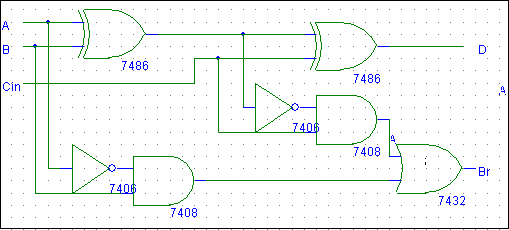
**Fig 9: Full adder circuit Diagram** Truth table for Fulladder



|  |  |  |  |
| --- | --- | --- | --- |
| A | B | D | B |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

**Fig 10: Half Subtractor circuit Diagram** Truth Table for Half subtractor

A full subtractor is 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. This circuit has three inputs and two outputs. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow, respectively.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Bin | D | Bout |
| 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 |

**Fig 11: Full subtractor circuit Diagram** Full subtractor truth table

#### Problem Statements

1. Write a VHDL code for Half adder, Full adder, Half subtractor and Full subtractor using data flow modeling and verify the output.
2. Write a VHDL code to realize a full adder using two half adders and a full subtractor using two half subtractor and verify the output.
3. Write a VHDL code to realize a full adder and full subtractor using behavioral modeling and verify the output.

VHDL Code:

#### Half adder using Data flow Modeling

Library ieee;

Library ieee.std\_logic\_1164.all; entity ha is

Port (a: in STD\_LOGIC; b : inSTD\_LOGIC;

s : out STD\_LOGIC; c : out STD\_LOGIC); endha;

architecture Behavioral of ha is begin

s <= a xorb ; c <= a and b ;

end Behavioral

#### Full adder using Data flow Modeling

Library ieee;

Library ieee.std\_logic\_1164.all; entity fa is

Port (a: in STD\_LOGIC; b : inSTD\_LOGIC;

Cin: in STD\_LOGIC; sum : out STD\_LOGIC; cout : out STD\_LOGIC); end fa;

architecture Behavioral of fa is begin

sum <= a xor b xorcin ;

cout<= ((a and b)or(b and c))or(a and c); end Behavioral

#### Half subtractor using data flow modeling

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOG IC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity hs is

Port ( a, b : in std\_logic; d, br : out std\_logic); end hs;

architecture dataflow of hs is begin

d<= a xor b;

br<= (not a) and b; end dataflow;

#### Full subtractor using Data flow modeling

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOG IC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity fs is

Port ( a, b, c : in std\_logic; d, br : out std\_logic);

end fs;

architecture dataflw of fs is begin

d<= a xor b xor c;

br<= ((not a) and (b xor c)) or (b and c); end datafolw;

#### Full adder using two half adder

ENTITY full\_adder IS PORT(A,B,Cin: IN std\_logic ; S, C : OUT std\_logic);

END full\_adder;

ARCHITECTURE str OF full\_adder IS

--component Declaration Component ha IS PORT(A,B: IN std\_logic ; S, Cout : OUT std\_logic); END Component;

signal s1,c2,c1:std\_logic;

BEGIN

X1: ha port map(A,B,s1,c1); X2: ha port map(s1,Cin,S,c2); C<=C1 or C2;

END str;

#### Full subtractor using Structural modeling

ENTITY full\_sub IS PORT(A,B,bin: IN std\_logic ; d, br : OUT std\_logic);

END full\_sub;

ARCHITECTURE str OF full\_sub IS

--component Declaration Component fs IS PORT(A,B: IN std\_logic ; d, b : OUT std\_logic); END Component;

signal s1,c2,c1:std\_logic;

BEGIN

X1: hs port map(A,B,s1,c1); X2: hs port map(s1,Cin,d,c2); br<=C1 or C2;

END str;

#### Full adder using Behavioral Modeling

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity fab is

Port ( a : in std\_logic; b : in std\_logic; c : instd\_logic;

s : out std\_logic; cr : out std\_logic);

end fab;

architecture Behavioral of fab is begin

process(a,b,c) begin

if(a='0' and b='0' and c='0')then s<='0';

cr<='0';

elsif( a='0' and b='0' andc='1')then s<='1';

cr<='0';

elsif( a='0' and b='1' andc='0')then s<='1';

cr<='0';

elsif( a='0' and b='1' andc='1')then s<='0';

cr<='1';

elsif( a='1' and b='0' andc='0')then s<='1';

cr<='0';

elsif( a='1' and b='0' andc='1')then s<='0';

cr<='1';

elsif( a='1' and b='1' andc='0')then s<='0';

cr<='1';

else s<='1';

cr<='1';

end if;

end process; end Behavioral;

**Full subtractor using behavioral Modeling**

ibraryieee;

use ieee.std\_logic\_1164.all; entity fullsub is

port(a:instd\_logic\_vector(2 downto 0); d,b:outstd\_logic);

end fullsub;

architecture fullsub1 of fullsub is begin

process(a) begin

if a="000" then d<='0';b<='0'; elsif a="001" then d<='1';b<='1'; elsif a="010" then d<='1';b<='1'; elsif a="011" then d<='0';b<='1'; elsif a="100" then d<='1';b<='0'; elsif a="101" then d<='0';b<='0'; elsif a="110" then d<='0';b<='0'; else d<='1';b<='1';

end if; end process;

end fullsub1;

#### Pre-lab questions:

1. State De-morgan’s theorem and mention its use.
2. Express Y=A+B+C in canonical sop form.
3. Define Minterm and Maxterm .

#### Post-lab questions:

1. Draw an active high tri-state Gate & write its truth table.
2. Show how to connect NAND gates to get an AND gate and OR gate?
3. Write a VHDL code for half adder using behavioral modelling and by using that block need to perform full adder circuit.

#### Result: