

**JK Lakshmipat University**

*Institute of Engineering & Technology*

**LAB1 (ASSIGNMENT SET -1)**

*CS1134: Computer Organisation and Architecture*

**Submitted by**  
Name: Siddhi Nyati  
Roll No: 2022btech101

**Faculty Supervisor:**

Dr. Pranab Roy

Dr. Ajai Jain

Activity no: 1

1.1: Half Adder

**Question 1:** Design a VHDL model using dataflow architecture for half adder.

**Theory:**

A Half Adder is a fundamental combinational logic circuit that adds two single-bit binary digits. It is composed of two primary gates: the XOR gate and the AND gate. The Half Adder takes two input bits, often denoted as ‘A’ and ‘B’, and produces two outputs: the ‘Sum’ and the ‘Carry’. The ‘Sum’ represents the result of the addition, while the ‘Carry’ indicates if there is an overflow into the next higher bit.

The Half Adder is limited to only two binary digits and cannot handle multi-bit numbers.

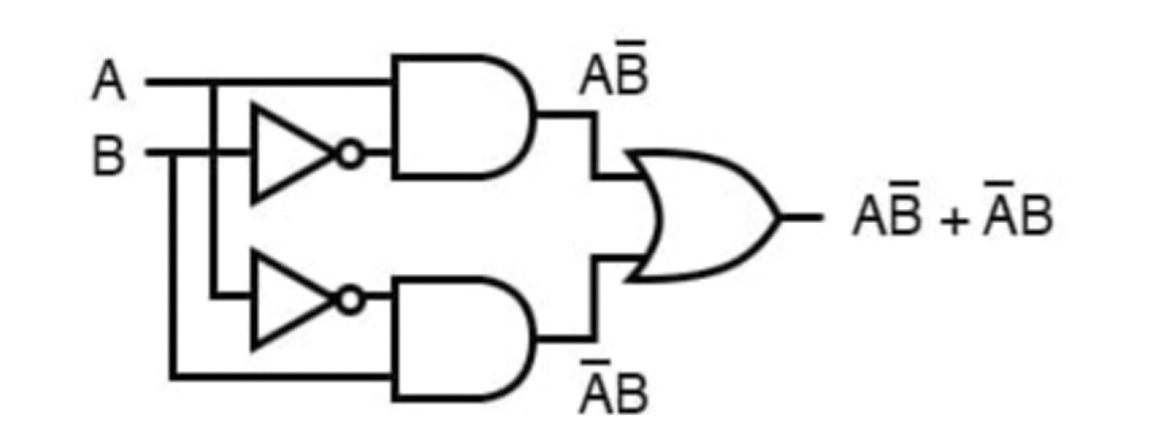
**Boolean Expressions for Half Adder:**

* **Sum:** Sum = A ⊕ B (which can also be represented as A+B)
* **Carry:** Carry = A . B (using the AND operation)

To better understand the behaviour of the Half Adder, we can analyze the truth tables for the XOR and AND gates.

A yellow and black logo

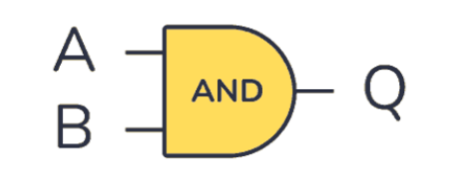
Description automatically generated



**XOR Gate Truth Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***A (Input)*** | ***B (Input)*** | ***AB′*** | ***A’B*** | ***Q (Output)*** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |

The Boolean expression for the XOR gate can be represented as: Q = (A . B’) + (A’ . B)



**AND Gate Truth Table:**

|  |  |  |
| --- | --- | --- |
| ***A (Input)*** | ***B (Input)*** | ***Q (Output) = AB*** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

By combining the outputs from the XOR and AND gates, we obtain the complete truth table for the Half Adder:

**Half Adder Truth Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***A (Input)*** | ***B (Input)*** | ***SUM (Output)*** | ***CARRY (Output)*** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

A diagram of a circuit

Description automatically generated

**Dataflow Modeling in VHDL**

VHDL, or Very High-Speed Integrated Circuit Hardware Description Language, is a powerful tool used for designing hardware systems. It enables the modeling of digital circuits at various levels of abstraction. VHDL supports three primary modeling styles: Dataflow, Behavioral, and Structural.

In this context, we will focus on Dataflow modeling. This approach describes the flow of data through a system without detailing the internal components. It emphasizes the relationships between inputs and outputs using logical operators such as AND, OR, and NOT.

The advantages of Dataflow modeling include its clarity, simplicity, and efficiency in representing digital systems, making it an excellent choice for designing circuits like the Half Adder.

**VHDL Code for Half Adder**

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 12:44:49 09/23/2024

-- Design Name:

-- Module Name: half\_adder - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity half\_adder is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

SUM : out STD\_LOGIC;

CARRY : out STD\_LOGIC);

end half\_adder;

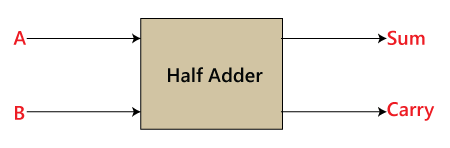
architecture dataflow of half\_adder is

begin

SUM <= A XOR B;

CARRY <= A AND B;

end dataflow;



**TEST BRANCH CODE FOR HALF ADDER**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity tb\_half\_adder is

end tb\_half\_adder;

architecture behavior of tb\_half\_adder is

-- Component declaration for the Unit Under Test (UUT)

component half\_adder

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

SUM : out STD\_LOGIC;

CARRY : out STD\_LOGIC);

end component;

signal A : STD\_LOGIC := '0';

signal B : STD\_LOGIC := '0';

signal SUM : STD\_LOGIC;

signal CARRY : STD\_LOGIC;

begin

-- Instantiate the Unit Under Test (UUT)

UUT: half\_adder Port map (

A => A,

B => B,

SUM => SUM,

CARRY => CARRY

);

-- Stimulus process

stim\_proc: process

begin

A <= '0'; B <= '0'; wait for 10 ns;

A <= '0'; B <= '1'; wait for 10 ns;

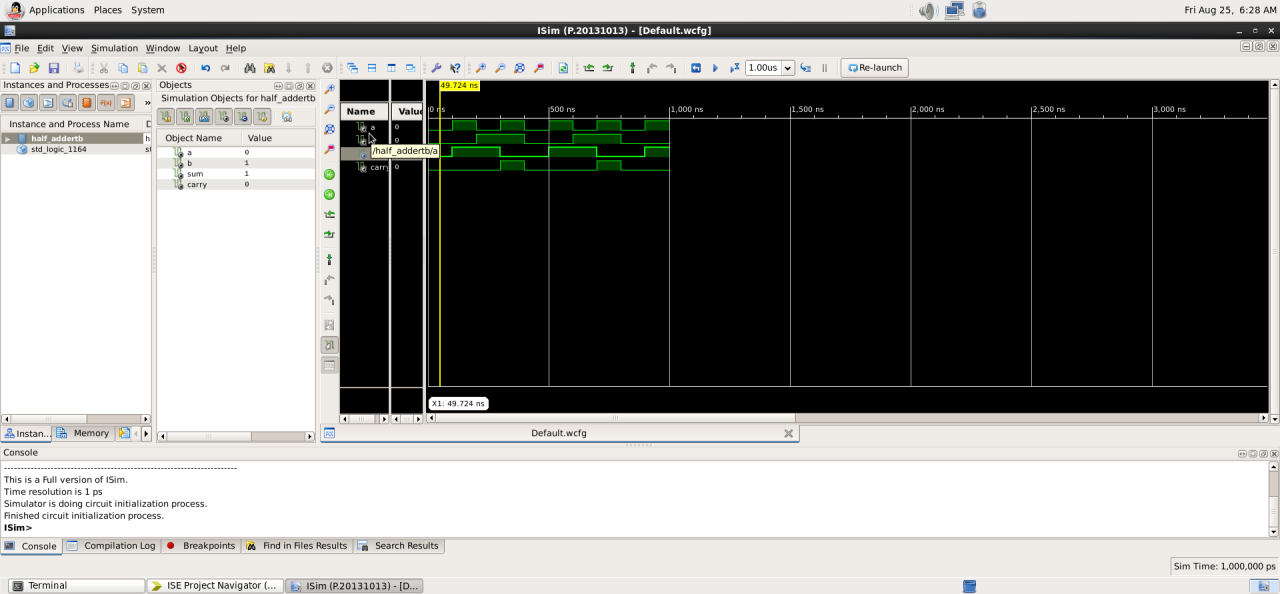
A <= '1'; B <= '0'; wait for 10 ns;

A <= '1'; B <= '1'; wait for 10 ns;

wait;

end process;

end behavior;

****