Experiment No. 4.1B

Design and implement a 2:1 MUX, 4:1 MUX and 8:1 MUX circuit in VHDL using Data Flow/ Behavioural Model.

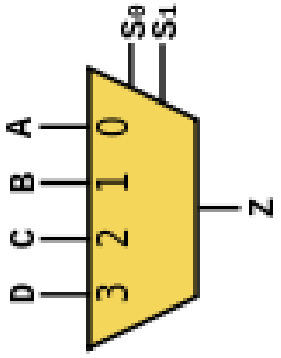
Objective:

Design and implement a 4:1 MUX circuit in VHDL using Behavioural Model.

Theory:

A multiplexer is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A mux has 2^n inputs with n selection lines and it gives one output. A mux is also called a data selector.

Circuit diagram:



Truth table:

Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

VHDL Source Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MUX_4_1 is
   Port ( a : in  STD_LOGIC ;
          b : in  STD_LOGIC ;
          c : in  STD_LOGIC ;
          d : in  STD_LOGIC ;
          s0 : in  STD_LOGIC ;
          s1 : in  STD_LOGIC ;
          z : out  STD_LOGIC);
end MUX_4_1;

architecture Behavioral of MUX_4_1 is
begin
process (A,B,C,D,S0,S1) is
          begin
                    if (S0 ='0' and S1 = '0') then
                    Z <= A;
                    elsif (S0 ='1' and S1 = '0') then
                     Z <= B;
                    elsif (S0 ='0' and S1 = '1') then
                    Z <= C;
                    else
                    Z <= D;
                    end if;

end process;
end Behavioral;
```

VHDL TestBench Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY TESTMUX4X1 IS
END TESTMUX4X1;

ARCHITECTURE behavior OF TESTMUX4X1 IS

   COMPONENT MUX_4_1
   PORT(
      A : IN  std_logic;
      B : IN  std_logic;
      C : IN  std_logic;
      D : IN  std_logic;
      S0 : IN  std_logic;
      S1 : IN  std_logic;
      Z : OUT  std_logic
```

```
    );
  END COMPONENT;

  signal A : std_logic := '0';
  signal B : std_logic := '0';
  signal C : std_logic := '0';
  signal D : std_logic := '0';
  signal S0 : std_logic := '0';
  signal S1 : std_logic := '0';

  signal Z : std_logic;

BEGIN

  uut: MUX_4_1 PORT MAP (
       A => A,
       B => B,
       C => C,
       D => D,
       S0 => S0,
       S1 => S1,
       Z => Z
      );

  stim_proc: process
  begin
   A <= '1';
   B <= '0';
   C <= '1';
   D <= '0';

   S0 <= '0';
   S1 <= '0';
   wait for 100 ns;
   S0 <= '1';
   S1 <= '0';
   wait for 100 ns;
   S0 <= '0';
   S1 <= '1';
   wait for 100 ns;
   S0 <= '0';
   S1 <= '1';
   wait for 100 ns;

  end process;

END;
```

Test Bench Waveform: