VHDL

Lab1

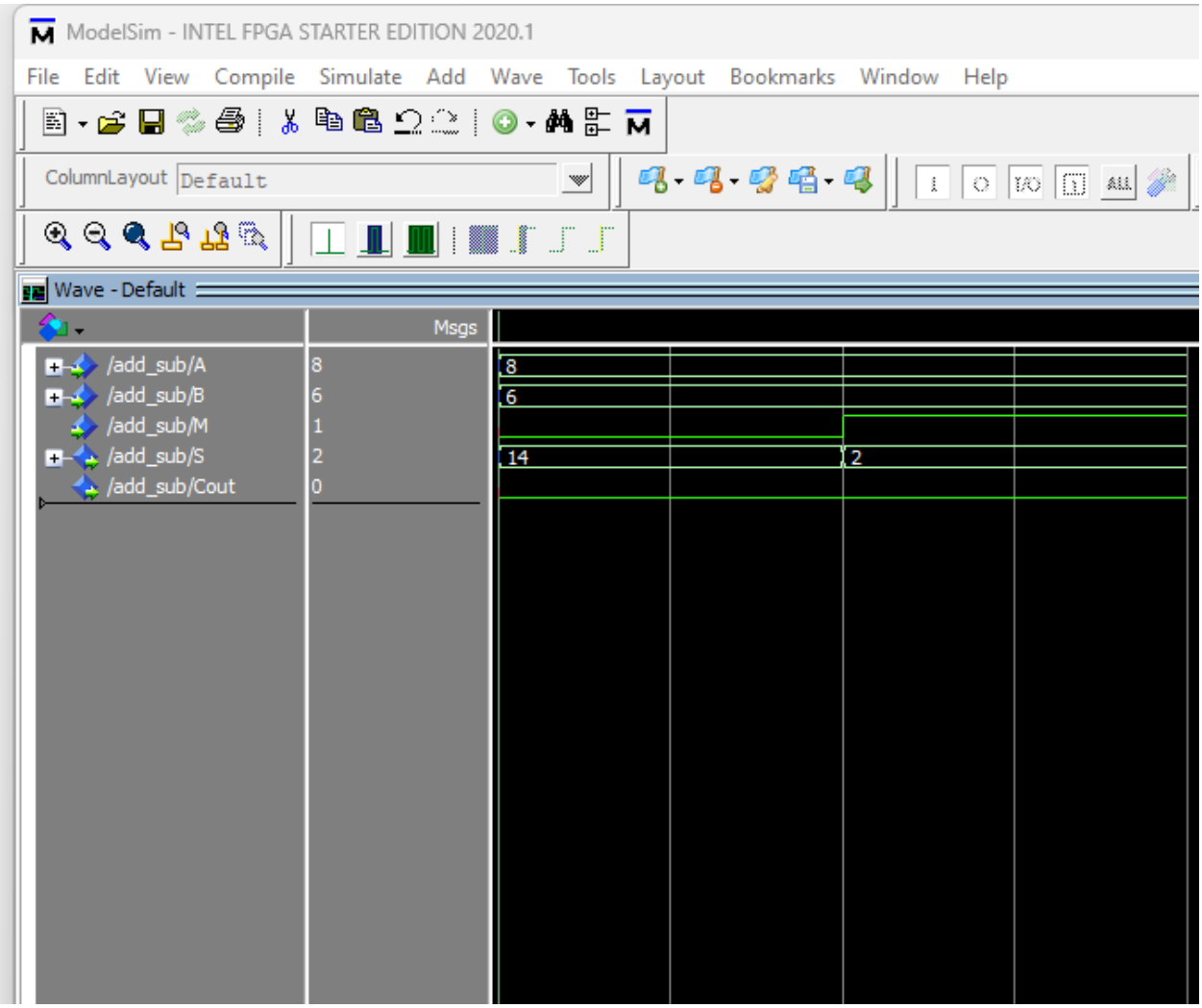
Name	Group
Mohamed Mabrouk Saad Dawod	2

1. Behavioral:
    a. Code:

```
LIBRARY IEEE;
LIBRARY WORK;
LIBRARY std;

USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_bit.ALL;
USE IEEE.std_logic_arith.all;
USE IEEE.numeric_std.all;
USE IEEE.std_logic_signed.all;

entity ADD_SUB is
   port (
      A, B: in  std_logic_vector (3 DOWNTO 0);
                        M: in std_logic;
      S: out std_logic_vector (3 downto 0);
      Cout: out std_logic
   );
end entity;

architecture behav of ADD_SUB is
begin
   process (A,B,M)
            variable result : std_logic_vector (4 downto 0);
   begin
                                    if (M ='0') then
         result := ('0' & A) + ('0' & B);
```

```
                    S <= result (3 downto 0);
            Cout <= result (4);
                else
                    result := ('0' & A) - ('0' & B);
                    S <= result (3 downto 0);
            Cout <= result (4);
                end if;
    end process;
end architecture;
```

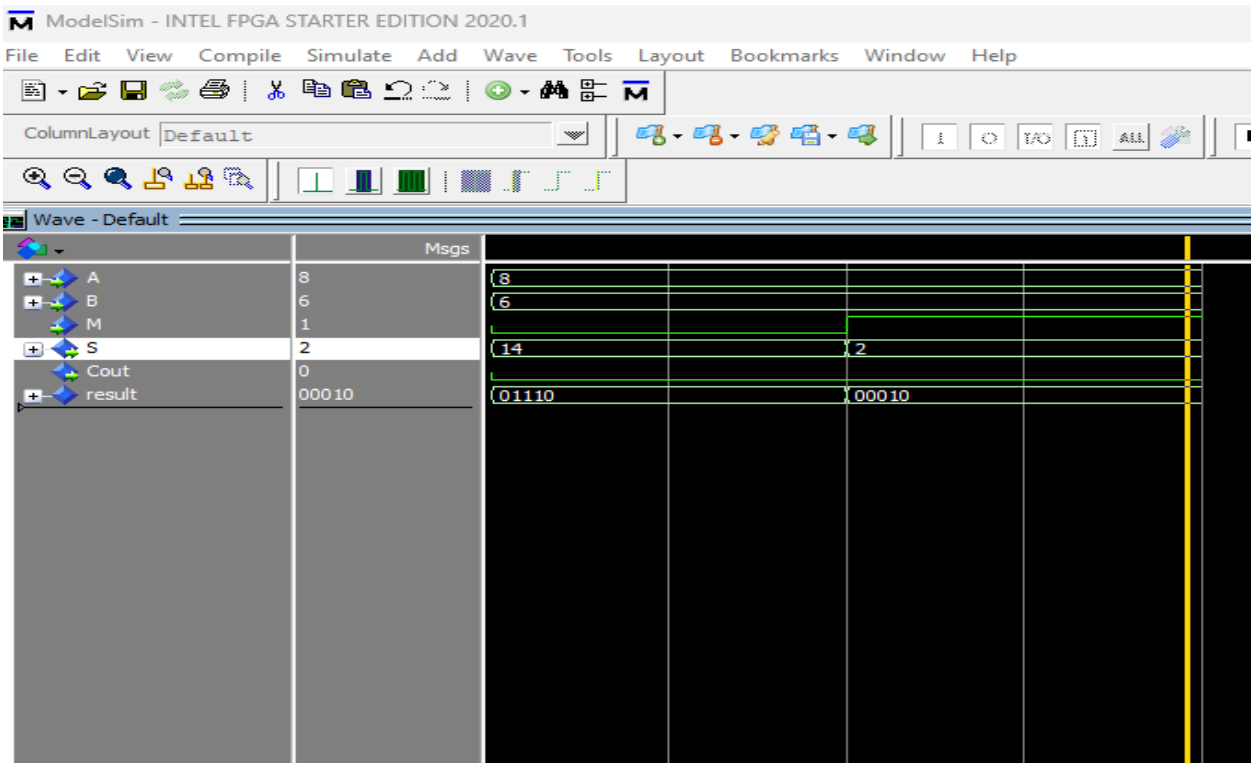
b. Sim:



2. Dataflow:
    a. Code:

```
LIBRARY IEEE;
LIBRARY WORK;
LIBRARY std;

USE IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_bit.ALL;
USE IEEE.std_logic_arith.all;
USE IEEE.numeric_std.all;
USE IEEE.std_logic_signed.all;

entity SUB_ADD_data_flow is
   port (
      A, B: in std_logic_vector(3 downto 0);
      M: in std_logic;
      S: out std_logic_vector(3 downto 0);
      Cout: out std_logic
   );
end entity;

architecture data_flow of SUB_ADD_data_flow is
                  signal result : std_logic_vector (4 downto 0);
begin
   result <= ('0' & A) + ('0' & B) when (M = '0') else ('0' & A) - ('0' & B);
         S <= result (3 downto 0);
         Cout <= result (4);
end architecture;
```

    b. Sim:

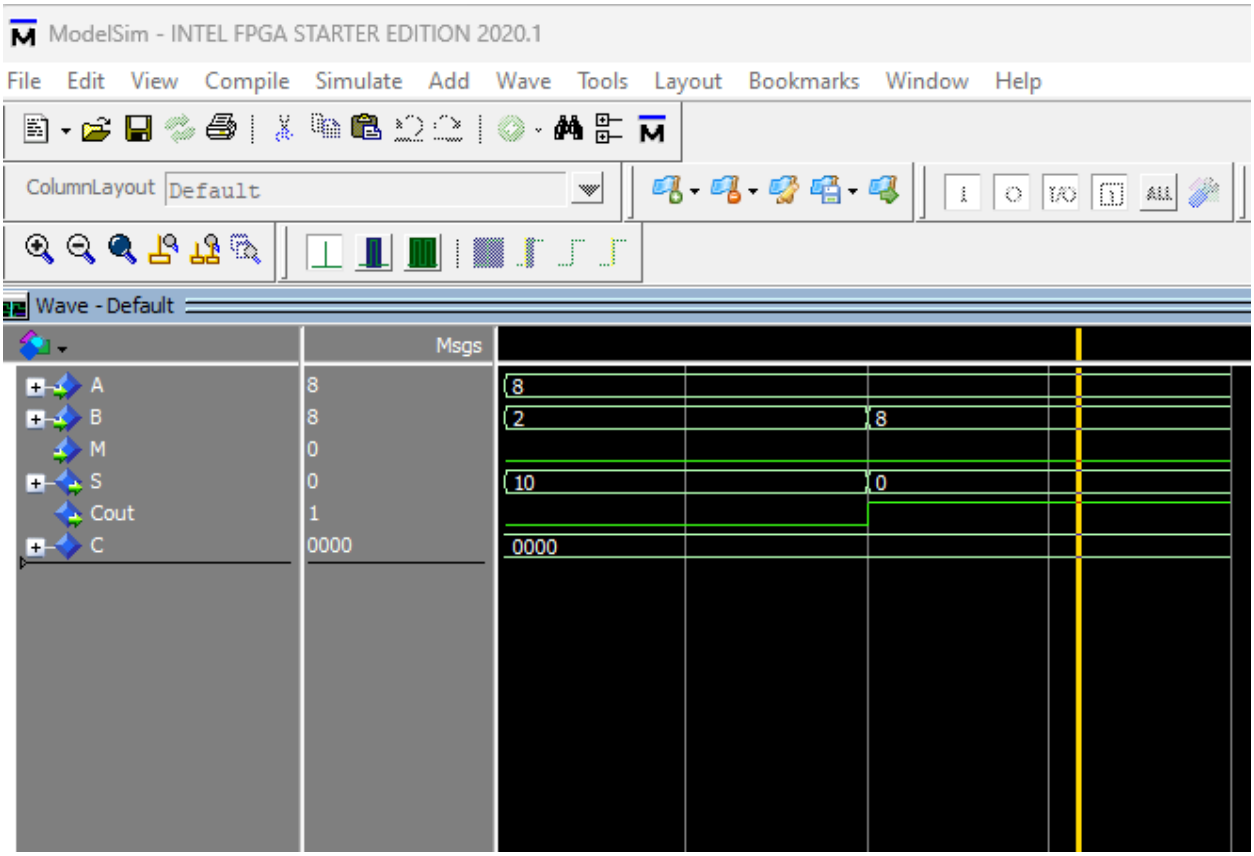


3. Structural:
    a. Code:

```
library ieee;
use ieee.std_logic_1164.all;

-- Entity declaration for half adder

entity HALF_ADDER is
   port (
      A, B : in  bit;
      Sum  : out bit;
      Cout: out bit
   );
end entity;

architecture behavior of HALF_ADDER is
begin
   Sum <= A xor B;
   Cout <= A and B;
end architecture;


--Entity of FULL ADDER

entity FULL_ADD_1bit is
   port (
      A, B, Cin : in  bit;
      Sum, Cout : out bit
   );
end entity;

architecture structural of FULL_ADD_1bit is
   component HALF_ADDER
      port (
         A, B : in  bit;
         Sum, Cout : out bit
      );
   end component;

   signal S1, C1, S2 : bit;
begin
   HALF_ADDER_1: HALF_ADDER port map (A => A, B => B, Sum => S1, Cout => C1);
   HALF_ADDER_2: HALF_ADDER port map (A => S1, B => Cin, Sum => Sum, Cout => S2);
   Cout <= C1 or S2;
end architecture;

--Entity of ADDER/SUBTRACTOR

entity SUB_ADD_struct is
   port (
```

```
        A, B: in bit_vector (3 downto 0);
        M: in bit;
        S: out bit_vector(3 downto 0);
        Cout: out bit
    );
end entity;

architecture structural of SUB_ADD_struct is
    component FULL_ADD_1bit is
        port (
            A, B, Cin: in bit;
            Sum, Cout: out bit
        );
    end component;

    signal C: bit_vector(3 downto 0);
begin
    FA0: FULL_ADD_1bit port map (A(0), B(0), M, S(0), C(0));
    FA1: FULL_ADD_1bit port map (A(1), B(1), C(0), S(1), C(1));
    FA2: FULL_ADD_1bit port map (A(2), B(2), C(1), S(2), C(2));
    FA3: FULL_ADD_1bit port map (A(3), B(3), C(2), S(3), Cout);
end architecture;
```

b. Sim:

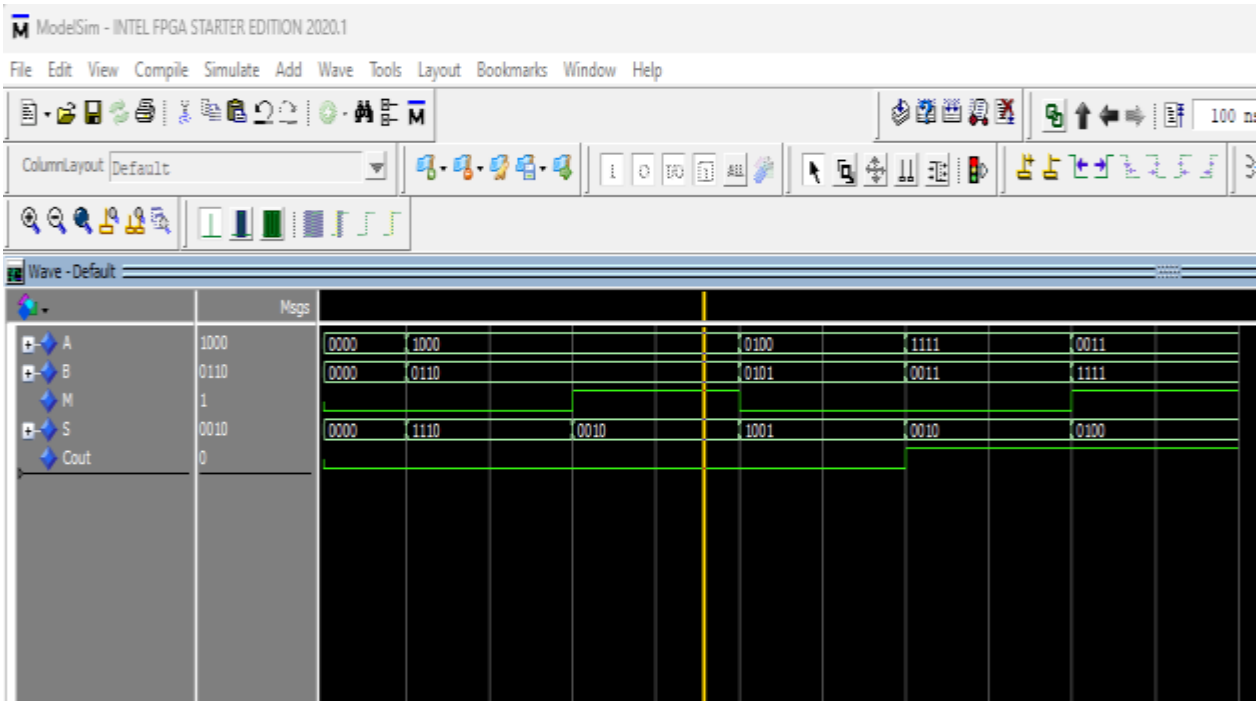


4. Testbench:
    a. Code:

```
LIBRARY IEEE;
LIBRARY WORK;
LIBRARY std;

USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_BIT.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.NUMERIC_STD.ALL;
USE IEEE.STD_LOGIC_SIGNED.ALL;
USE STD.TEXTIO.ALL;

entity ADD_SUB_TB is
end entity;

architecture tb_arch of ADD_SUB_TB is
   component ADD_SUB is
      port (
         A, B: in  std_logic_vector (3 DOWNTO 0);
         M: in std_logic;
         S: out std_logic_vector (3 downto 0);
         Cout: out std_logic
      );
   end component;

         --test bench signals

   signal A, B: std_logic_vector(3 downto 0);
   signal M: std_logic;
   signal S: std_logic_vector(3 downto 0);
   signal Cout: std_logic;

   constant clock_period: time := 10 ns;

begin

   --entity instantiation
   uut: ADD_SUB
      port map (
         A => A,
         B => B,
         M => M,
         S => S,
         Cout => Cout
      );

   stim_proc: process

         file input : text open read_mode is ("E:\Verilog\Quartus\VHDL\inputs.txt");
```

```
        file output : text open read_mode is ("E:\Verilog\Quartus\VHDL\outputs.txt");
        variable in_l,out_l : line;
        variable delay : time;
        VARIABLE s: string (1 TO 51);
        variable A_f,B_f : std_logic_vector (3 downto 0);
        variable M_f : std_logic;
        variable S_f : std_logic_vector (3 downto 0);
        variable Cout_f : std_logic;

  begin

                A <= "0000"; B <= "0000"; M <= '0'; wait for clock_period;
    while not endfile (input) loop

                readline (input, in_l);
                read (in_l,A_f);
                read (in_l,B_f);
                read (in_l,M_f);

                A <= A_f; B <= B_f; M <= M_f;  wait for clock_period;

                --S_f := S;                          commented lines (71/72/86/94) gives an
error (don't know why)
                --Cout_f := Cout;

                write (out_l,string'("in1= "));
                write (out_l, A_f);

                write (out_l,string'("in2= "));
                write (out_l, B_f);

                write (out_l,string'("in3= "));
                write (out_l, M_f);

                write (out_l,string'("Time="));
                write (out_l, NOW);

                --writeline (output, out_l);

                write (out_l,string'("out= "));
                write (out_l, S_f);

                write (out_l,string'("out= "));
                write (out_l, Cout_f);

                --writeline (output, out_l);

                wait for clock_period;
                end loop;
    wait;
  end process;
end architecture;
```

b. Sim:



c. Content of input file: