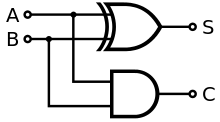
EXPERIMENT 2

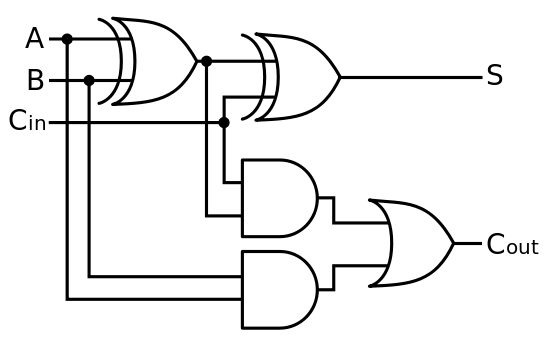
**AIM:** To design and simulate Half Adder and full Adder Using VHDL in ModelSim.

**Theory:** The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an overflow into the next digit of a multi-digit addition.

The half adder adds two input bits and generates a carry and sum, which are the two outputs of a half adder. The input variables of a half adder are called the augend and addend bits. The output variables are the sum and carry. The truth table and diagram for the half adder is:

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **SUM (S)** | **CARRY(C)** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

A **full adder** adds binary numbers and accounts for values carried in as well as out. A one-bit full-adder adds three one-bit numbers, often written as *A*, *B*, and *C*in; *A* and *B* are the operands, and *C*in is a bit carried in from the previous less-significant stage. The truth table and diagram for the Full adder is:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **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 |

**VHDL CODE: FOR HALF ADDER**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all;

ENTITY HALF\_ADDER IS

port (A, B:in std\_logic; SUM, CARRY: out std\_logic);

END ENTITY HALF\_ADDER;

--

ARCHITECTURE HALF\_ADDER\_STRUCTURAL OF HALF\_ADDER IS

component AND\_GATE

port (A, B:in std\_logic; Z: out std\_logic);

end component;

component XOR\_GATE

port (A, B:in std\_logic; Z: out std\_logic);

end component;

BEGIN

A1: AND\_GATE port map (A, B, CARRY);

X1: XOR\_GATE port map (A, B, SUM);

END ARCHITECTURE HALF\_ADDER\_STRUCTURAL;

ARCHITECTURE HALF\_ADDER\_BEHAV OF HALF\_ADDER IS

BEGIN

ha: process (a, b)

begin

if a = '1' then

sum <= not b;

CARRY <= b;

else

sum <= b;

carry <= '0';

end if;

end process ha;

END ARCHITECTURE HALF\_ADDER\_BEHAV;

ARCHITECTURE HALF\_ADDER\_DATAFLOW OF HALF\_ADDER IS

BEGIN

SUM<=A xor B;

CARRY<=A and B;

END ARCHITECTURE HALF\_ADDER\_DATAFLOW;

**FOR FULL ADDER**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all;

ENTITY FULL\_ADDER IS

port (A, B, CARRY\_IN: in std\_logic; SUM, CARRY: out std\_logic);

END ENTITY FULL\_ADDER;

--

ARCHITECTURE FULL\_ADDER\_DATAFLOW OF FULL\_ADDER IS

BEGIN

SUM <= A xor B xor CARRY\_IN;

CARRY <= (A and B) or (CARRY\_IN and A) or (CARRY\_IN and B);

END ARCHITECTURE FULL\_ADDER\_DATAFLOW;

ARCHITECTURE FULL\_ADDER\_BEHAV OF FULL\_ADDER IS

SIGNAL s: std\_logic\_vector (2 downto 0);

BEGIN

process (A, B, CARRY\_IN, s)

BEGIN

s (2) <=A; s (1) <= B; s (0) <= CARRY\_IN;

case s is

when "000" => SUM<='0'; CARRY<='0';

when "001" => SUM<='1'; CARRY<='0';

when "010" => SUM<='1'; CARRY<='0';

when "011" => SUM<='0'; CARRY<='1';

when "100" => SUM<='1'; CARRY<='0';

when "101" => SUM<='0'; CARRY<='1';

when "110" => SUM<='0'; CARRY<='1';

when "111" => SUM<='1'; CARRY<='1';

when others => null;

end case;

end process;

END ARCHITECTURE FULL\_ADDER\_BEHAV;

ARCHITECTURE FULL\_ADDER\_STRUCTURAL OF FULL\_ADDER IS

component AND\_GATE

port (A,B:in std\_logic;Z: out std\_logic);

end component;

component XOR\_GATE

port (A,B:in std\_logic;Z: out std\_logic);

end component;

component OR\_GATE

port(A,B:in std\_logic;Z: out std\_logic);

end component;

SIGNAL var1,var2,var3: std\_logic;

BEGIN

X1: XOR\_GATE port map (A,B,var1);

X2: XOR\_GATE port map (CARRY\_IN,var1,SUM);

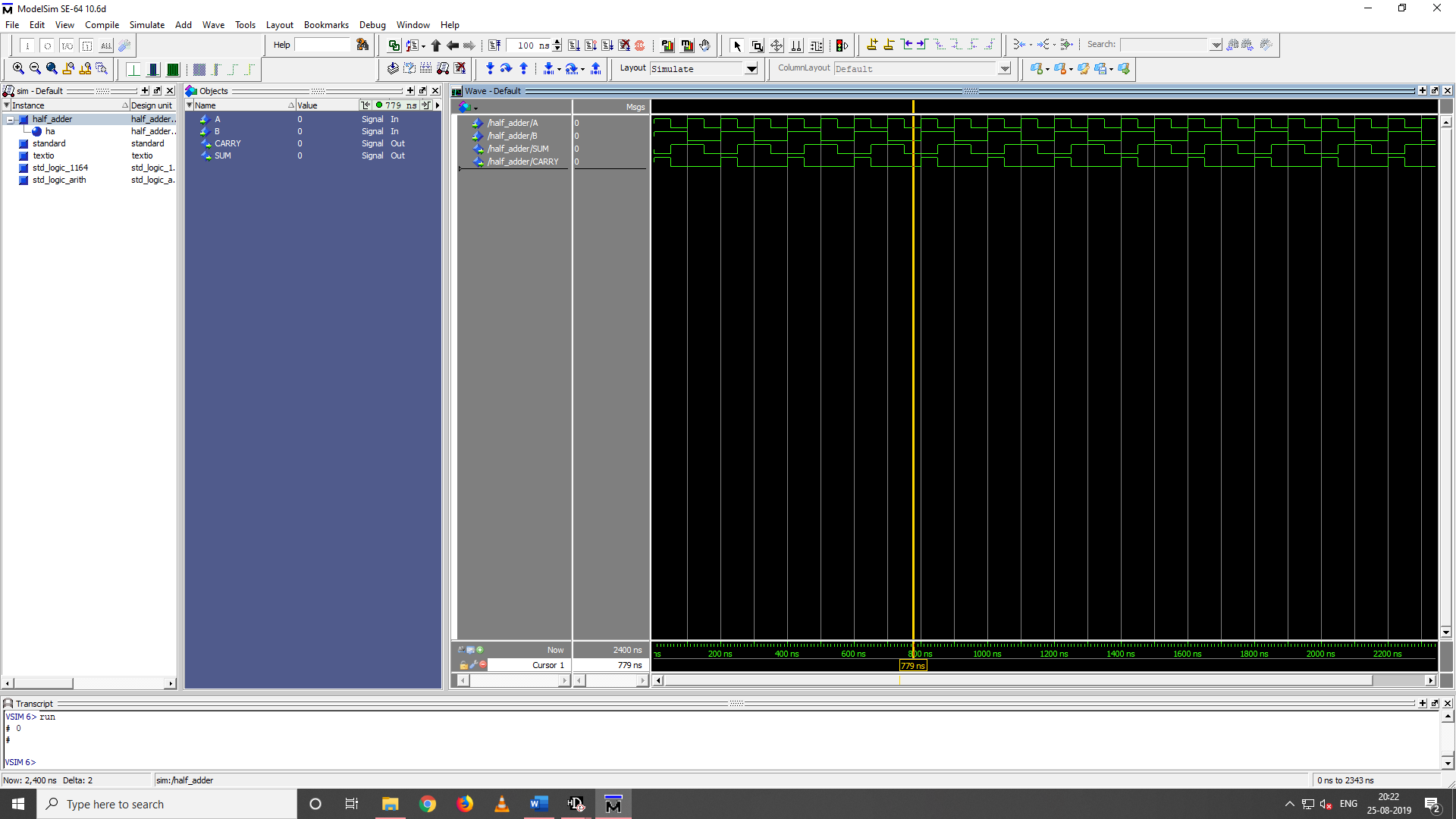
A1: AND\_GATE port map (var1,CARRY\_IN,var2);

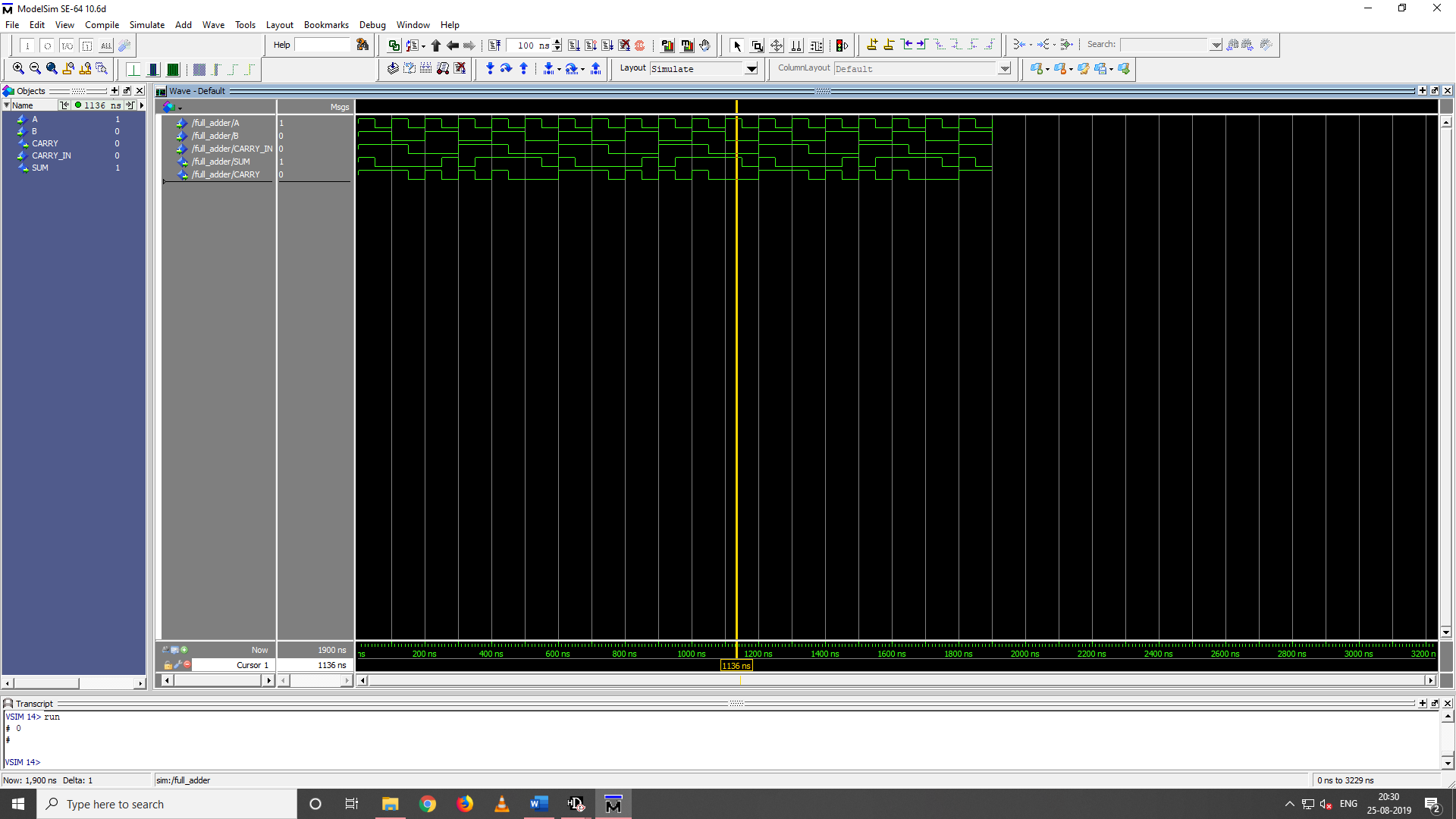
A2: AND\_GATE port map (A,B,var3);

O1: OR\_GATE port map (var2,var3,CARRY);

END ARCHITECTURE FULL\_ADDER\_STRUCTURAL;

**OUTPUTS:**



**HALF ADDER**

**FULL ADDER**

**RESULT: Half-adder and Full-adder are simulated using model sim and HDL designer.**