MEHRAN UNIVERSITY OF ENGINEERING AND TECHNOLOGY, JAMSHORO
DEPARTMENT OF ELECTRONIC ENGINEERING

FPGA Based Digital Design (ES-373)
Batch: 20ES (6th Semester)

Lab Experiment #05
Designing of Combinational Logic Circuits (Full-Adder, Comparator & Multiplexer)

Name	KARAN	**Roll #**	20ES062
Signature of Lab Tutor		**Date**	

RUBRICS:

Performance Metric	TEAMWORK	PARTICIPATION	CONDUCTING EXPERIMENT	USE OF MODERN ENGINEERIN G TOOLS	DATA ANALYSIS	CALCULATION AND CODING	OBSERVATION /PROGRAM RESULTS	**Total Score**
	0.5	**0.5**	**1**	**1**		**1**	**1**	**05**
Obtained								

OBJECTIVE(S)

The purpose of this lab is to:

#	Topic	# Of Lectures	CLO	Taxonomy level
1	Design & Synthesize Combinational Logic Circuits using Xilinx ISE Software.	3	4,5	P3, A4
2	Test the circuits through simulation using ISim and implementation on Digilent FPGA boards.			

OUTCOME(S)

a. An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.	**PL-5**: Modern Tool Usage
b. An ability to communicate effectively	**PLO10**: Communication

LAB REQUIREMENTS:

- Standard PC with Xilinx ISE Design Suite 12.3 or latest, Software installed.
- Digilent Adept Software.
- **NEXYS2** Spartan 3E (XC3S500K or XC3S1200K) board.

DISCUSSION:

A **full adder** is a logical circuit that performs an addition operation on three one-bit binary numbers often written as a, b, and c_{in}. The full adder produces a two-bit output sum typically represented with the signals C_{out} and S. Truth Table for a 1-Bit Full Adder is given below.

a	b	Cin	sum	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

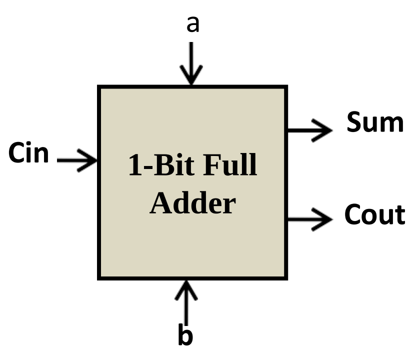


Figure 1: Full Adder Truth Table, and Signal Block

A full adder can be implemented in many ways such as with a custom transistor-level circuit or composed of other gates. One example implementation is with

$$\boldsymbol{Sum = A \oplus B \oplus C_{in}}$$
$$\boldsymbol{C_{out} = A.B + (C_{in}.(A \oplus B)C_{in})}$$

DESIGNING PROCEDURE:

DESIGN ENTRY

1. Invoke Xilinx ISE Design Suite Software.
   "Select **File > New Project"**
2. Enter the Project name **full_adder** in the Name field. Verify that **HDL** is selected as the Top-Level Source Type, and click on **NEXT** again click on **NEXT** and Click on **Finish.**
3. **Now Create a New Source file**; Go to **Project** > **New Source**.
   - ✓ Select **VHDL Module** and Enter the source file name **full_adder**.
   - ✓ Click on **Next**. Enter the Input and output Ports name and click **NEXT** and click **Finish.**

VHDL Code for the Full adder is

```
LIBRARY ieee;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY full_adder  is
   Port (a, b, Cin : in STD_LOGIC;
   Sum, Cout : out  STD_LOGIC;
END full_adder;
ARCHITECTURE Behavioral of full_adder IS
BEGIN
sum <= (a XOR b) XOR Cin;
Cout <= (a AND b) OR (Cin AND (a XOR b));
END Behavioral;
```

Simulation

After Creating project and new Source File now it is ready to simulate the Design. To simulate the Full Adder follow the following steps.

1. In the Design Panel, select the **Simulation** radio button.
2. Select full_adder Behavioral file and double click on Simulator Behavioral Model in the Process window.
3. Select the Simulation from the menu bar and click on Restart.

4. Force the input signal (a, b, cin).
5. Select the Simulation from the menu bar and click on Run. It shows the following Simulation result of Full Adder.

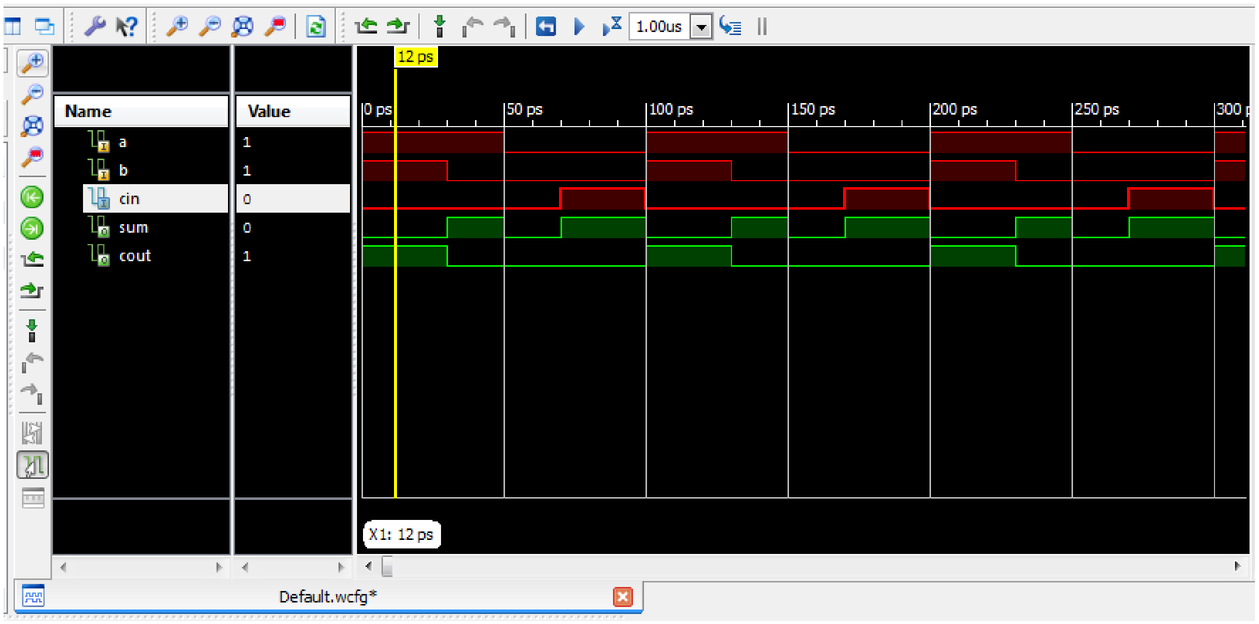


Figure: 2. Simulation Result for Full adder

Synthesis

1. To synthesize the design, double click on the **Synthesize Design** option in the **Processes window.**
2. Now the schematic diagram of the Full adder can be viewed by double clicking View RTL Schematic under Synthesize-XST menu in the Process Window, (Figure 1.3).
3. Similarly, you can view the Technology Schematic Symbol by double clicking **View Technology Schematic** under Synthesize-XST menu in the Process Window, (Figure 1.4). This view provides insights into how the synthesized logic maps to the specific **resources** of the FPGA technology.

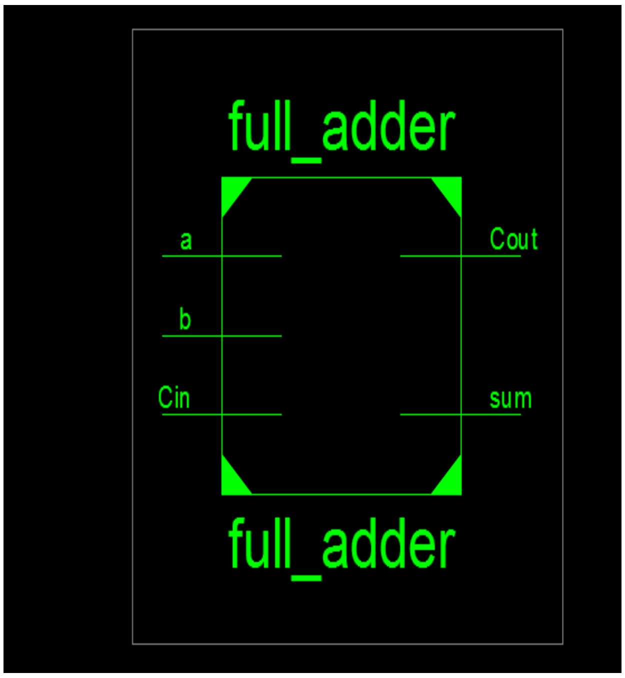


Figure 3: Schematic diagram for Full adder

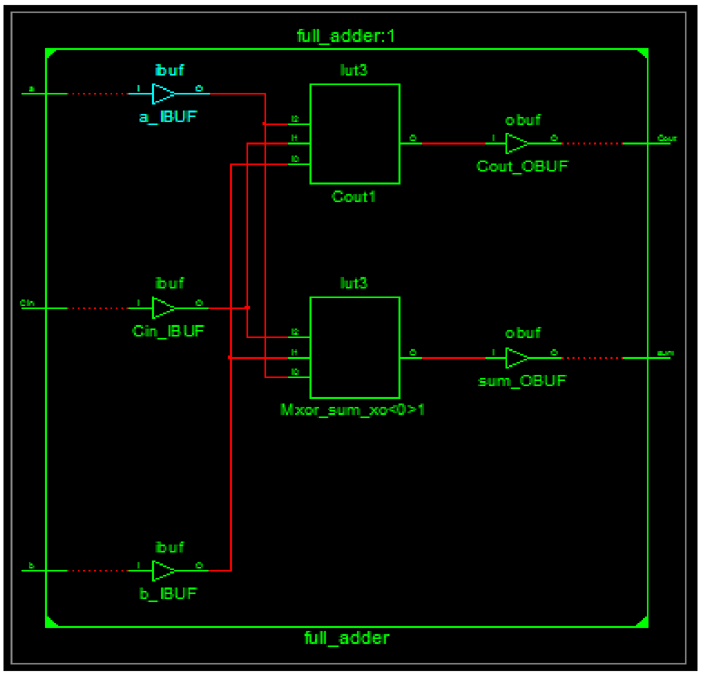
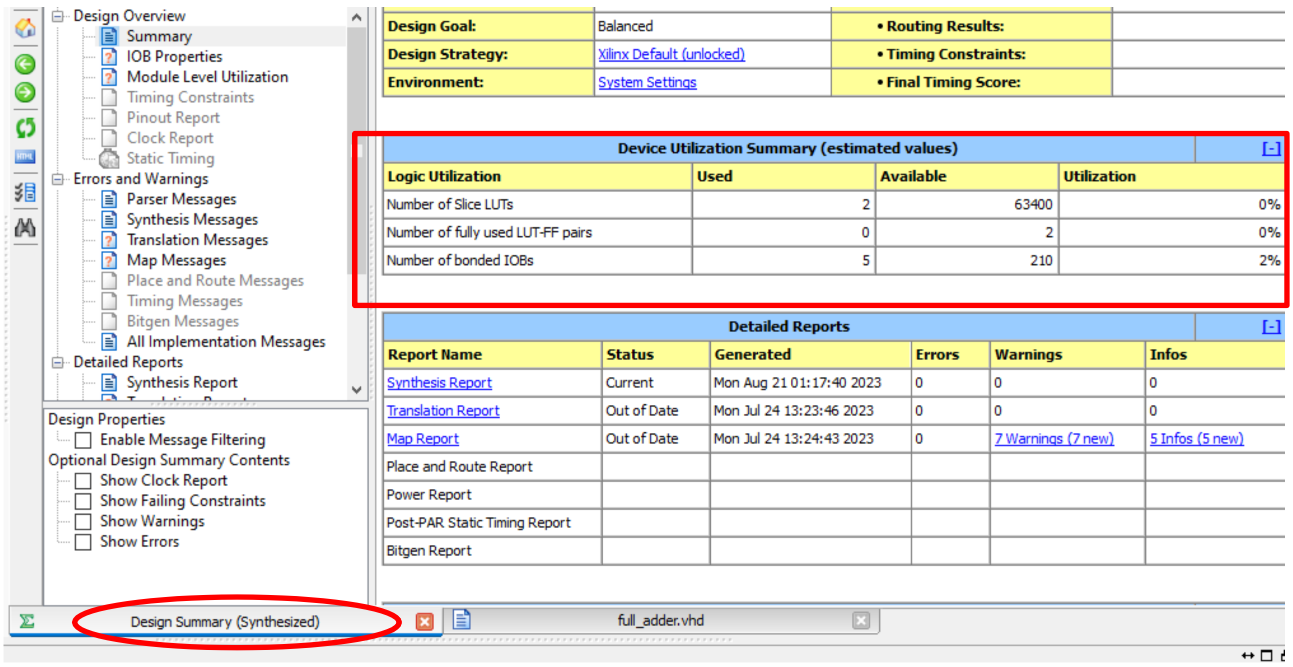


Figure 4: Technology Schematic Symbol for Full

5. Additionally, you can gain a comprehensive understanding of your design's resource utilization by checking the "Device Utilization Summary." This report provides detailed information about the utilization of various FPGA resources such as lookup tables (LUTs), flip-flops, and more. It's a valuable resource for assessing the efficiency of your design and making any necessary optimizations. To access the Device Utilization Summary, click on Design summary (Synthesized) as shown in figure 5.



- Design Overview
  - Summary
  - IOB Properties
  - Module Level Utilization
  - Timing Constraints
  - Pinout Report
  - Clock Report
  - Static Timing
- Errors and Warnings
  - Parser Messages
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Implementation Messages
- Detailed Reports
  - Synthesis Report

Design Properties
- [ ] Enable Message Filtering

Optional Design Summary Contents
- [ ] Show Clock Report
- [ ] Show Failing Constraints
- [ ] Show Warnings
- [ ] Show Errors

Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	2	63400	0%
Number of fully used LUT-FF pairs	0	2	0%
Number of bonded IOBs	5	210	2%

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon Aug 21 01:17:40 2023	0	0	0
Translation Report	Out of Date	Mon Jul 24 13:23:46 2023	0	0	0
Map Report	Out of Date	Mon Jul 24 13:24:43 2023	0	7 Warnings (7 new)	5 Infos (5 new)
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

Figure 5: Device Utilization Summary

Implementation

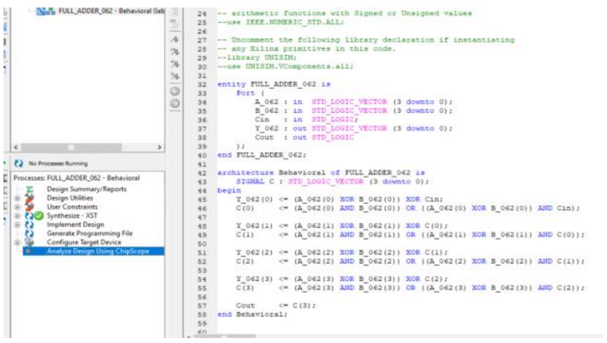
1. Before implement the design you must create the User Constraint File **UCF**.
2. After creating the UCF file Single Click on full_adder.ucf file from within Project Navigator, and then select “Edit Constraints (Text)” from the Process window.
3. Assigning the pins to inputs and outputs of Full adder as following and save Ucf file.

```
NET "a" LOC = "G18";
NET "b" LOC = "H18";
NET "Cin" LOC = "K18";

NET "sum” LOC = "J14";
NET "Cout" LOC = "J15";
```

4. After assigning the Pins double click on “Implement Design” option in the **Processes window**. It will go through steps like **Translate, Map and Place & Route.**
5. Now create a programming file (bit stream file) of the design. This is done by clicking once on your top-level design in the Sources Pane, followed by a double click on “**Generate Programming File**” in the process window.
6. Once the programming file (bit stream file) is generated, the file has to be downloaded to the **NEXYS2 Spartan3E** device.

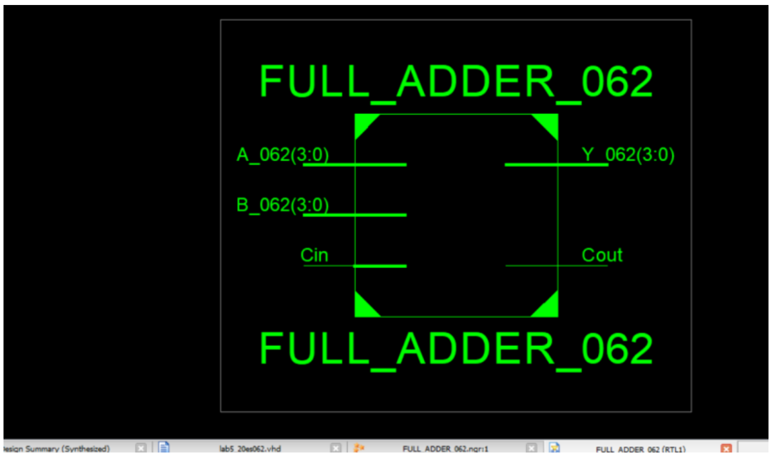
4-Bit Full Adder :-

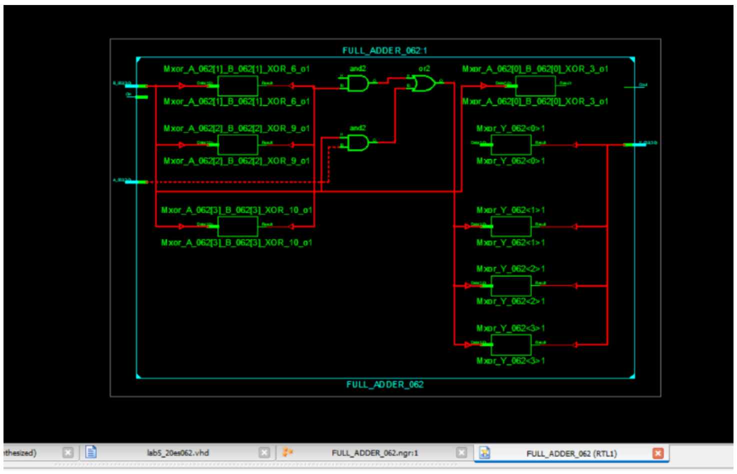
VHDL code



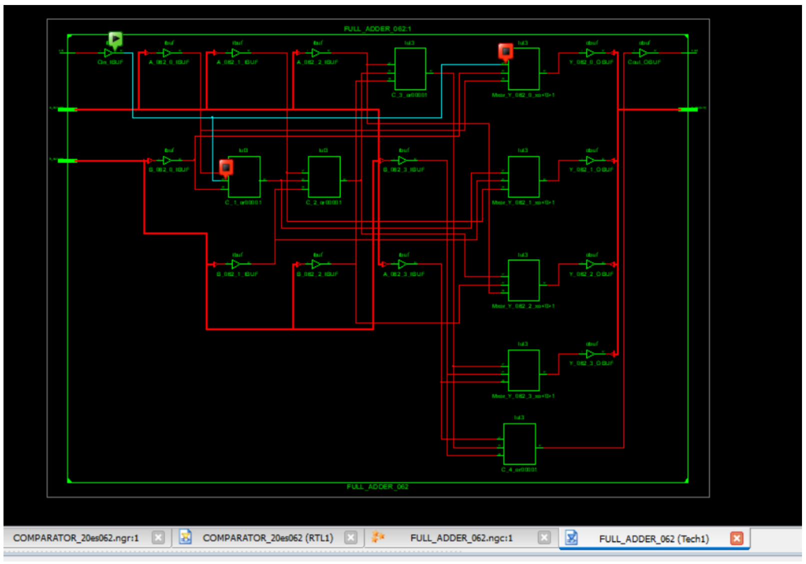
```vhdl
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity FULL_ADDER_062 is
    Port (
        A_062 : in  STD_LOGIC_VECTOR (3 downto 0);
        B_062 : in  STD_LOGIC_VECTOR (3 downto 0);
        Cin   : in  STD_LOGIC;
        Y_062 : out STD_LOGIC_VECTOR (3 downto 0);
        Cout  : out STD_LOGIC
    );
end FULL_ADDER_062;

architecture Behavioral of FULL_ADDER_062 is
    SIGNAL C : STD_LOGIC_VECTOR (3 downto 0);
begin
    Y_062(0) <= (A_062(0) XOR B_062(0)) XOR Cin;
    C(0)     <= (A_062(0) AND B_062(0)) OR ((A_062(0) XOR B_062(0)) AND Cin);

    Y_062(1) <= (A_062(1) XOR B_062(1)) XOR C(0);
    C(1)     <= (A_062(1) AND B_062(1)) OR ((A_062(1) XOR B_062(1)) AND C(0));

    Y_062(2) <= (A_062(2) XOR B_062(2)) XOR C(1);
    C(2)     <= (A_062(2) AND B_062(2)) OR ((A_062(2) XOR B_062(2)) AND C(1));

    Y_062(3) <= (A_062(3) XOR B_062(3)) XOR C(2);
    C(3)     <= (A_062(3) AND B_062(3)) OR ((A_062(3) XOR B_062(3)) AND C(2));

    Cout     <= C(3);
end Behavioral;
```

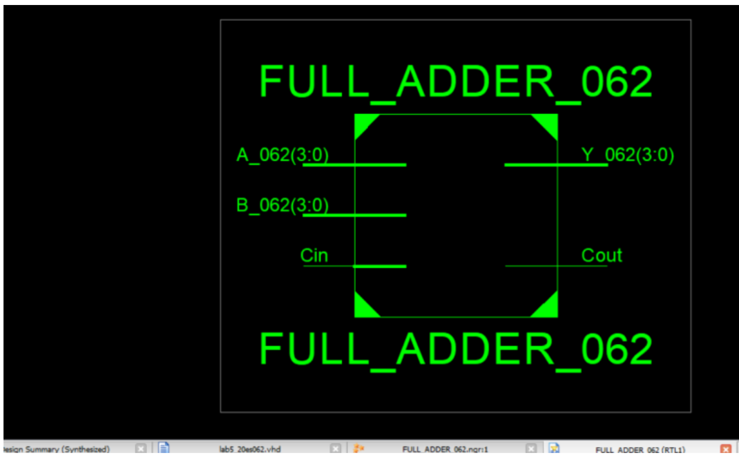
RTL



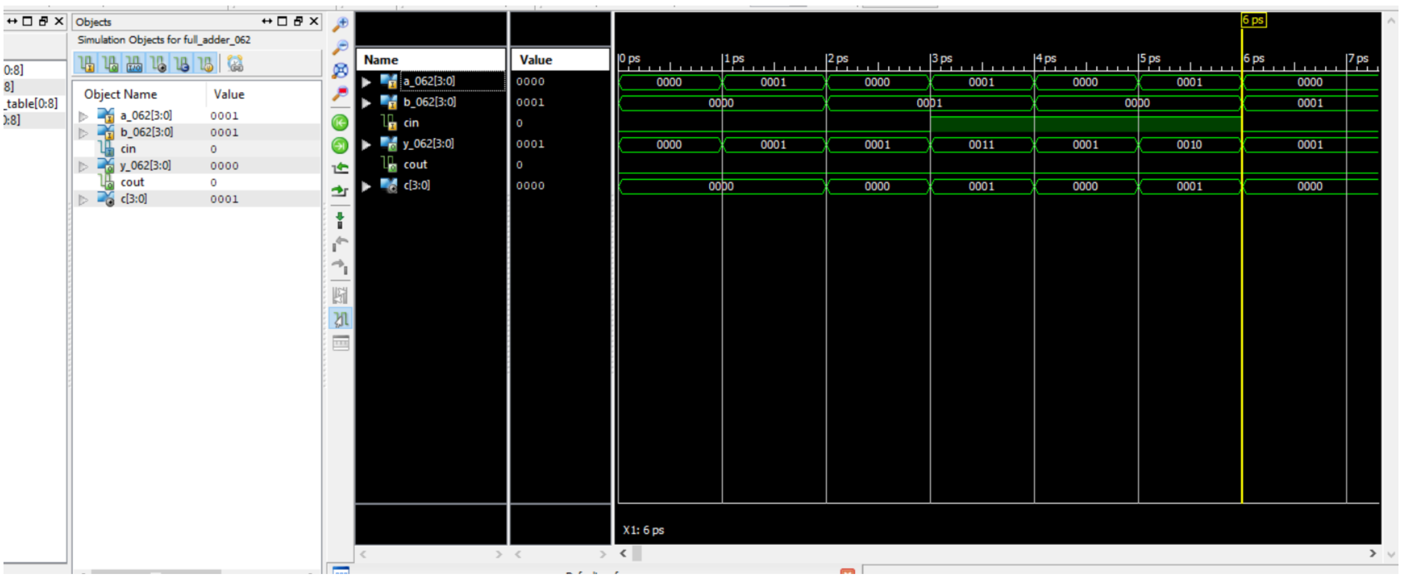


Schematic





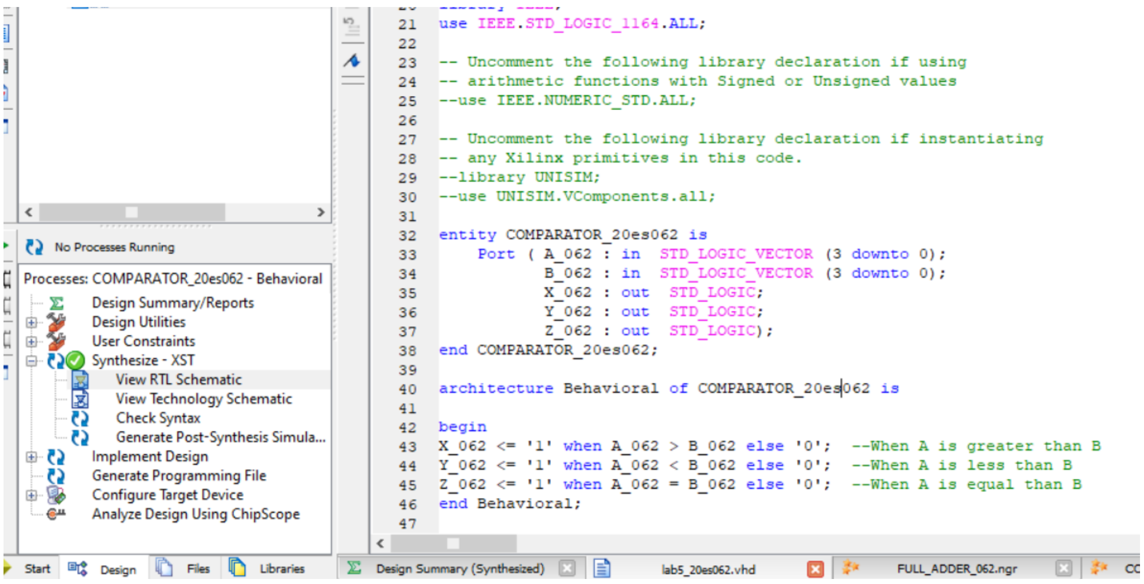
Simulation


Simulation Objects for full_adder_062

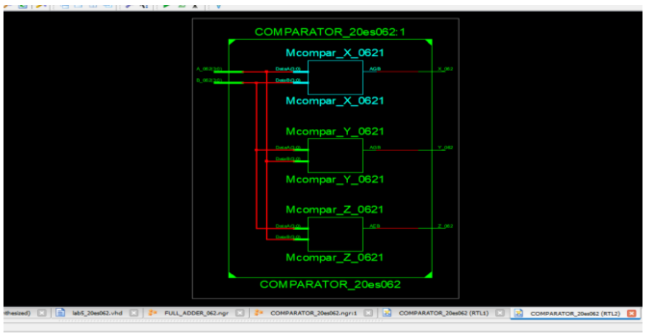
Object Name	Value
a_062[3:0]	0001
b_062[3:0]	0001
cin	0
y_062[3:0]	0000
cout	0
c[3:0]	0001

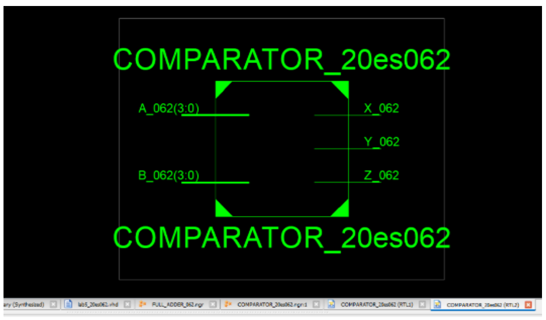
Name	Value
a_062[3:0]	0000
b_062[3:0]	0001
cin	0
y_062[3:0]	0001
cout	0
c[3:0]	0000

4-BIT COMPARATOR

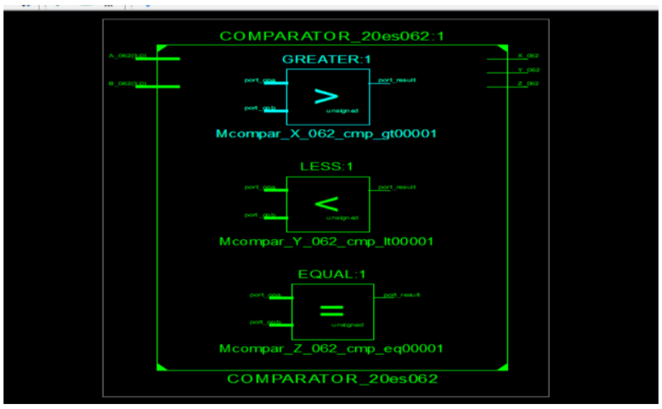

```
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity COMPARATOR_20es062 is
    Port ( A_062 : in  STD_LOGIC_VECTOR (3 downto 0);
           B_062 : in  STD_LOGIC_VECTOR (3 downto 0);
           X_062 : out  STD_LOGIC;
           Y_062 : out  STD_LOGIC;
           Z_062 : out  STD_LOGIC);
end COMPARATOR_20es062;

architecture Behavioral of COMPARATOR_20es062 is

begin
X_062 <= '1' when A_062 > B_062 else '0';   --When A is greater than B
Y_062 <= '1' when A_062 < B_062 else '0';   --When A is less than B
Z_062 <= '1' when A_062 = B_062 else '0';   --When A is equal than B
end Behavioral;
```

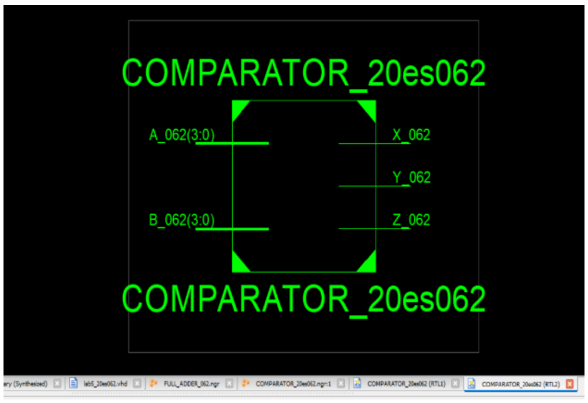
schematic



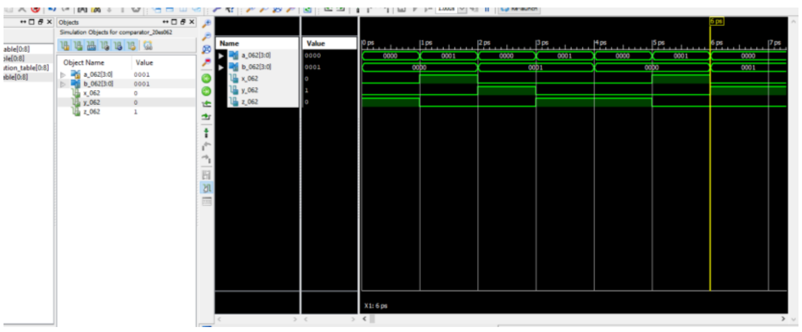


RTL :-

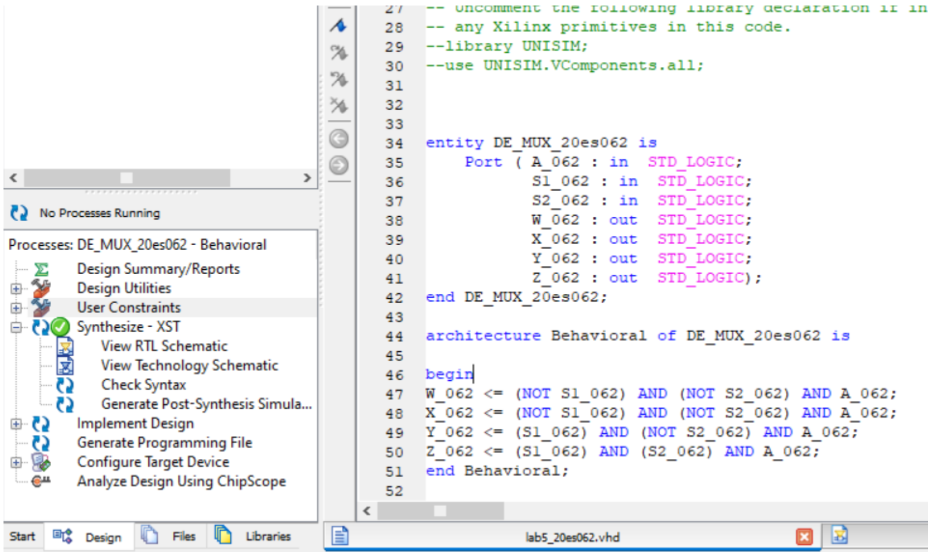

COMPARATOR_20es062:1
GREATER:1
>
Mcompar_X_062_cmp_gt00001
LESS:1
<
Mcompar_Y_062_cmp_lt00001
EQUAL:1
=
Mcompar_Z_062_cmp_eq00001
COMPARATOR_20es062



Simulation

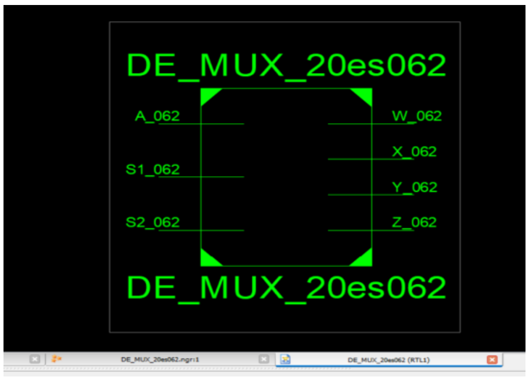


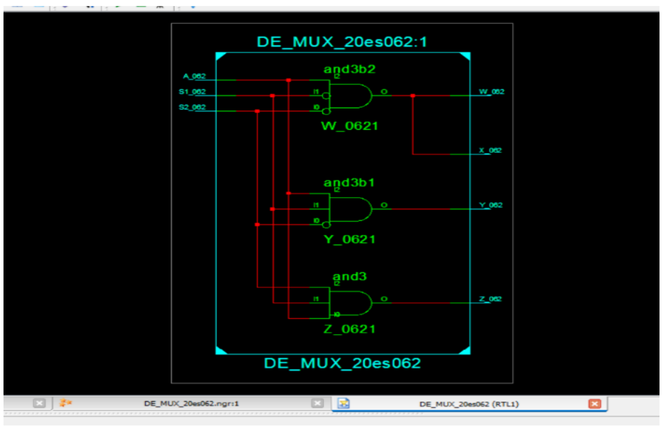
DE_MUX
VHDL code



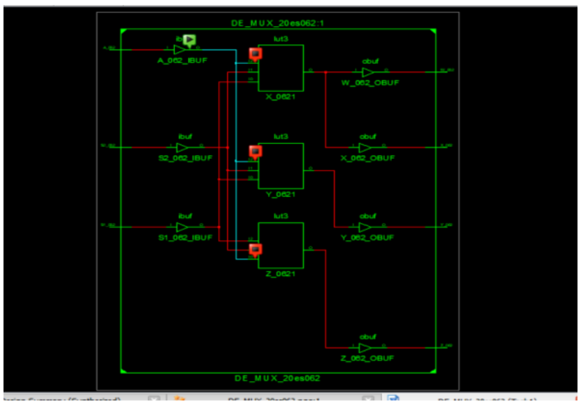
```
-- uncomment the following library declaration if in
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;



entity DE_MUX_20es062 is
    Port ( A_062 : in  STD_LOGIC;
           S1_062 : in  STD_LOGIC;
           S2_062 : in  STD_LOGIC;
           W_062 : out  STD_LOGIC;
           X_062 : out  STD_LOGIC;
           Y_062 : out  STD_LOGIC;
           Z_062 : out  STD_LOGIC);
end DE_MUX_20es062;

architecture Behavioral of DE_MUX_20es062 is

begin
W_062 <= (NOT S1_062) AND (NOT S2_062) AND A_062;
X_062 <= (NOT S1_062) AND (NOT S2_062) AND A_062;
Y_062 <= (S1_062) AND (NOT S2_062) AND A_062;
Z_062 <= (S1_062) AND (S2_062) AND A_062;
end Behavioral;

```

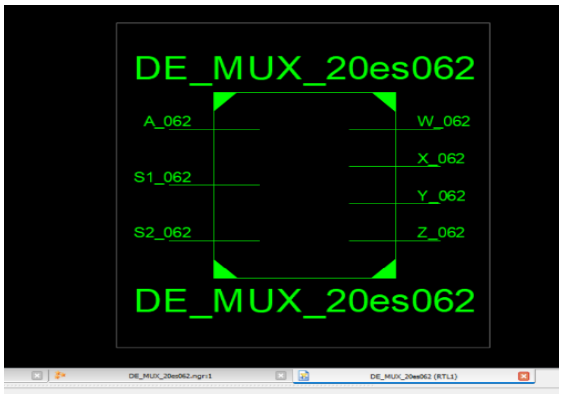
RTL



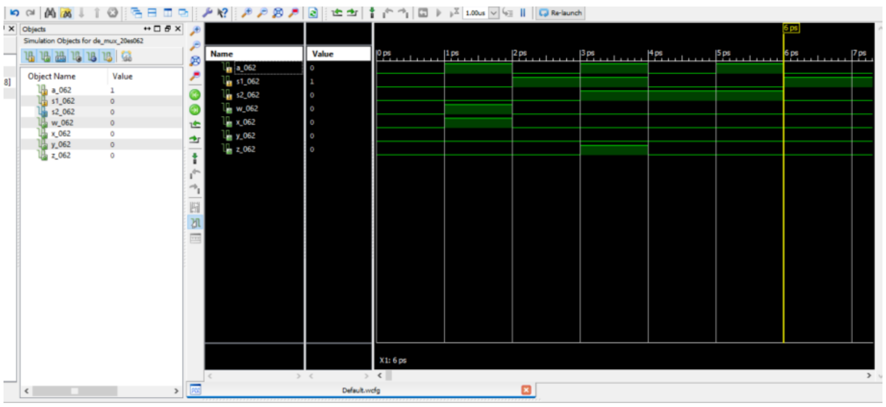


Schematic





simulation


Objects
Simulation Objects for de_mux_20es062

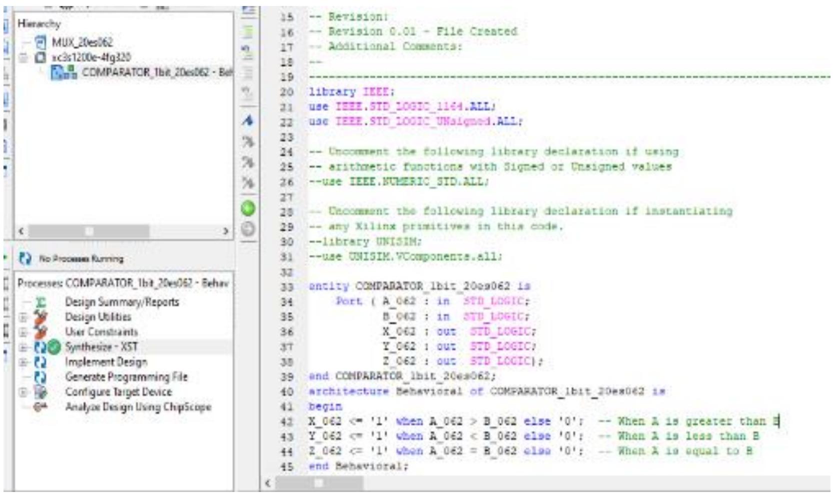
Object Name	Value
a_062	1
s1_062	0
s2_062	0
w_062	0
x_062	0
y_062	0
z_062	0

Name	Value
a_062	0
s1_062	1
s2_062	0
w_062	0
x_062	0
y_062	0
z_062	0

Review Questions

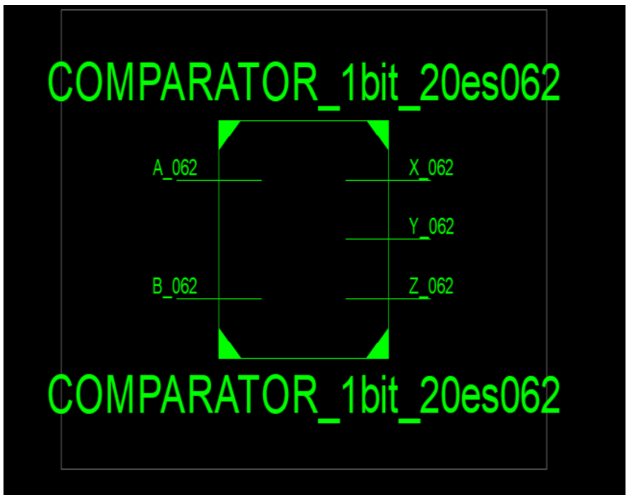
1 bit comparator

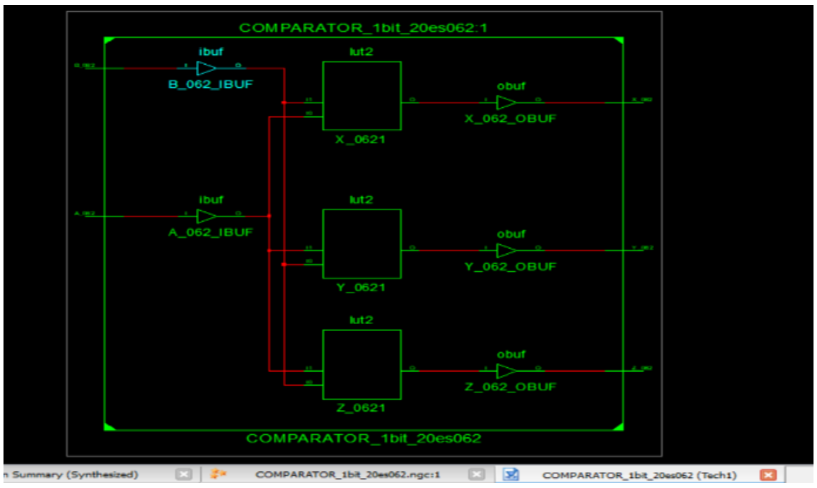
```
entity COMPARATOR_1bit_20es062 is
    Port ( A_062 : in  STD_LOGIC;
           B_062 : in  STD_LOGIC;
           X_062 : out  STD_LOGIC;
           Y_062 : out  STD_LOGIC;
           Z_062 : out  STD_LOGIC);
end COMPARATOR_1bit_20es062;
architecture Behavioral of COMPARATOR_1bit_20es062 is
begin
X_062 <= '1' when A_062 > B_062 else '0';
Y_062 <= '1' when A_062 < B_062 else '0';
Z_062 <= '1' when A_062 = B_062 else '0';
end Behavioral;
```

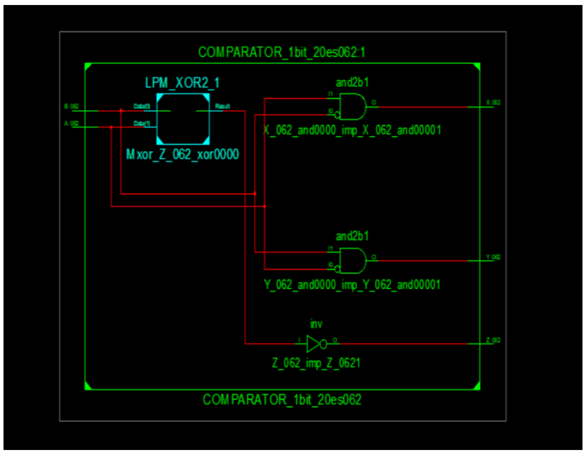


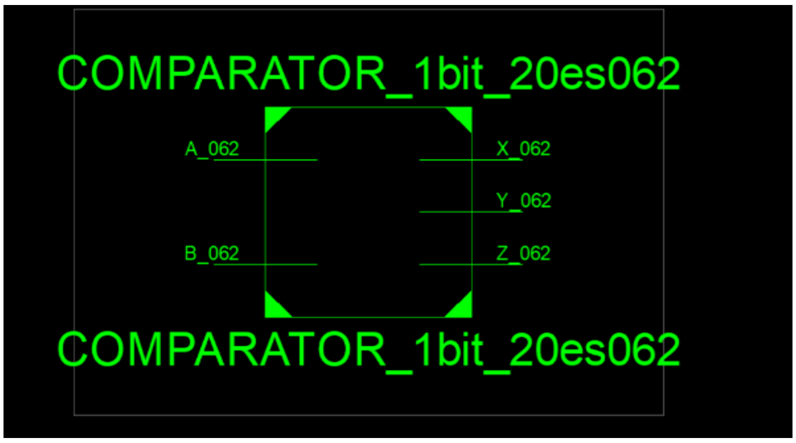
```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNsigned.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity COMPARATOR_1bit_20es062 is
    Port ( A_062 : in  STD_LOGIC;
           B_062 : in  STD_LOGIC;
           X_062 : out  STD_LOGIC;
           Y_062 : out  STD_LOGIC;
           Z_062 : out  STD_LOGIC);
end COMPARATOR_1bit_20es062;
architecture Behavioral of COMPARATOR_1bit_20es062 is
begin
X_062 <= '1' when A_062 > B_062 else '0';   -- When A is greater than B
Y_062 <= '1' when A_062 < B_062 else '0';   -- When A is less than B
Z_062 <= '1' when A_062 = B_062 else '0';   -- When A is equal to B
end Behavioral;
```

RTL and schematic

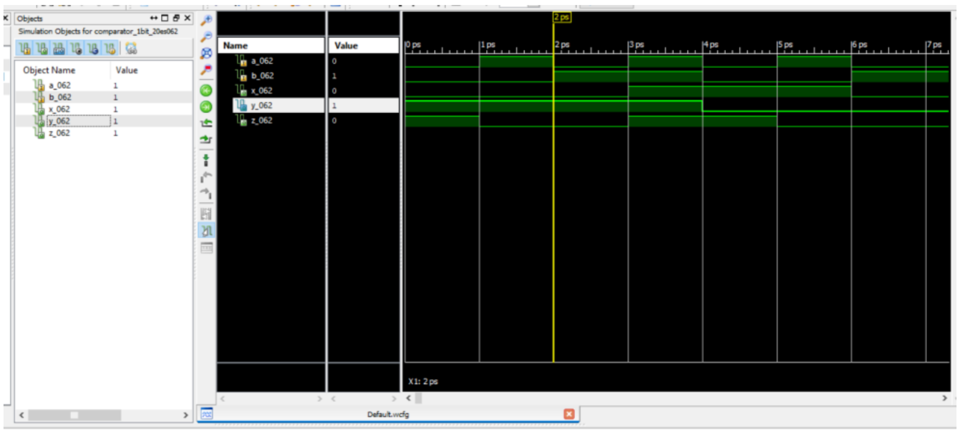





COMPARATOR_1bit_20es062:1
LPM_XOR2_1
Mxor_Z_062_xor0000
and2b1
X_062_and0000_imp_X_062_and00001
and2b1
Y_062_and0000_imp_Y_062_and00001
inv
Z_062_imp_Z_0621
COMPARATOR_1bit_20es062

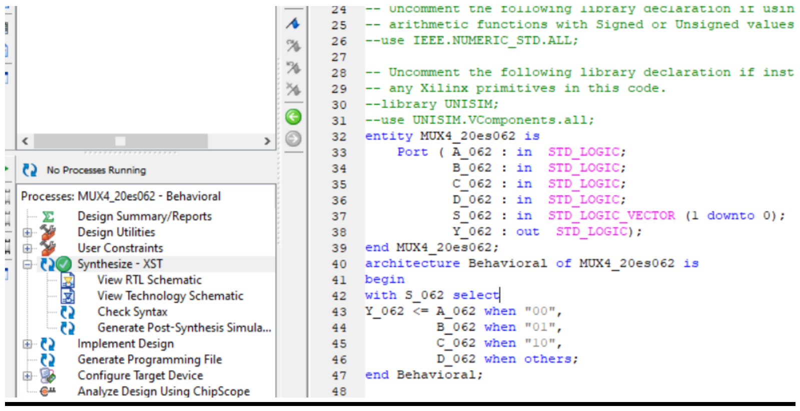


Simulation :-



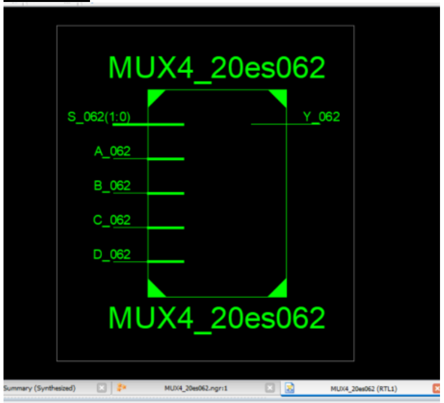
4 input multiplexer

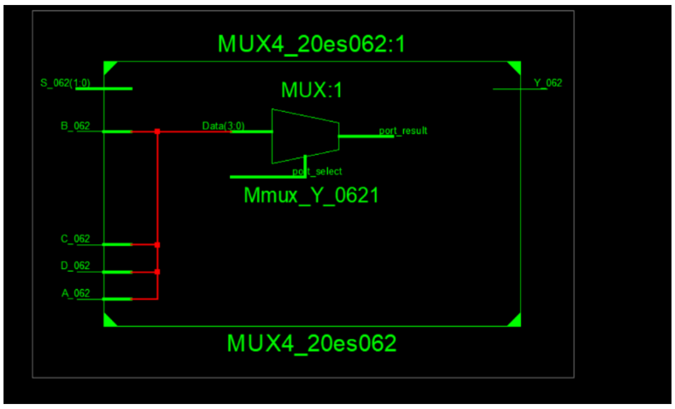
```
entity MUX4_20es062 is
    Port ( A_062 : in  STD_LOGIC;
           B_062 : in  STD_LOGIC;
           C_062 : in  STD_LOGIC;
           D_062 : in  STD_LOGIC;
           S_062 : in  STD_LOGIC_VECTOR (1 downto 0);
           Y_062 : out  STD_LOGIC);
end MUX4_20es062;
architecture Behavioral of MUX4_20es062 is
begin
with S_062 select
Y_062 <= A_062 when "00",
         B_062 when "01",
         C_062 when "10",
         D_062 when others;
end Behavioral;
```



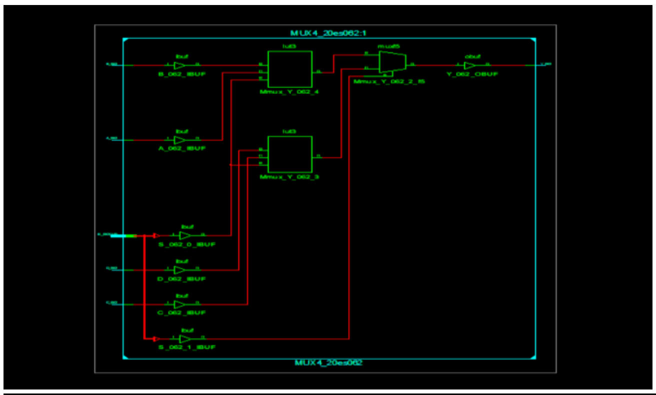
```
-- Uncomment the following library declaration if usin
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if inst
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity MUX4_20es062 is
    Port ( A_062 : in  STD_LOGIC;
           B_062 : in  STD_LOGIC;
           C_062 : in  STD_LOGIC;
           D_062 : in  STD_LOGIC;
           S_062 : in  STD_LOGIC_VECTOR (1 downto 0);
           Y_062 : out  STD_LOGIC);
end MUX4_20es062;
architecture Behavioral of MUX4_20es062 is
begin
with S_062 select
Y_062 <= A_062 when "00",
         B_062 when "01",
         C_062 when "10",
         D_062 when others;
end Behavioral;
```

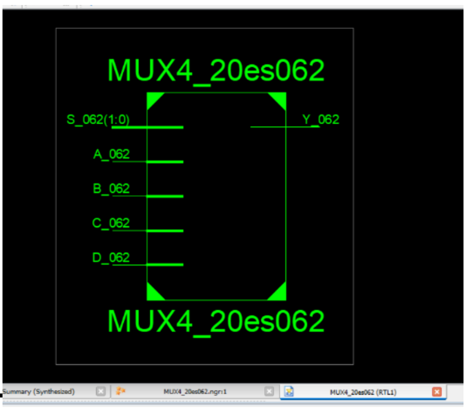
RTL



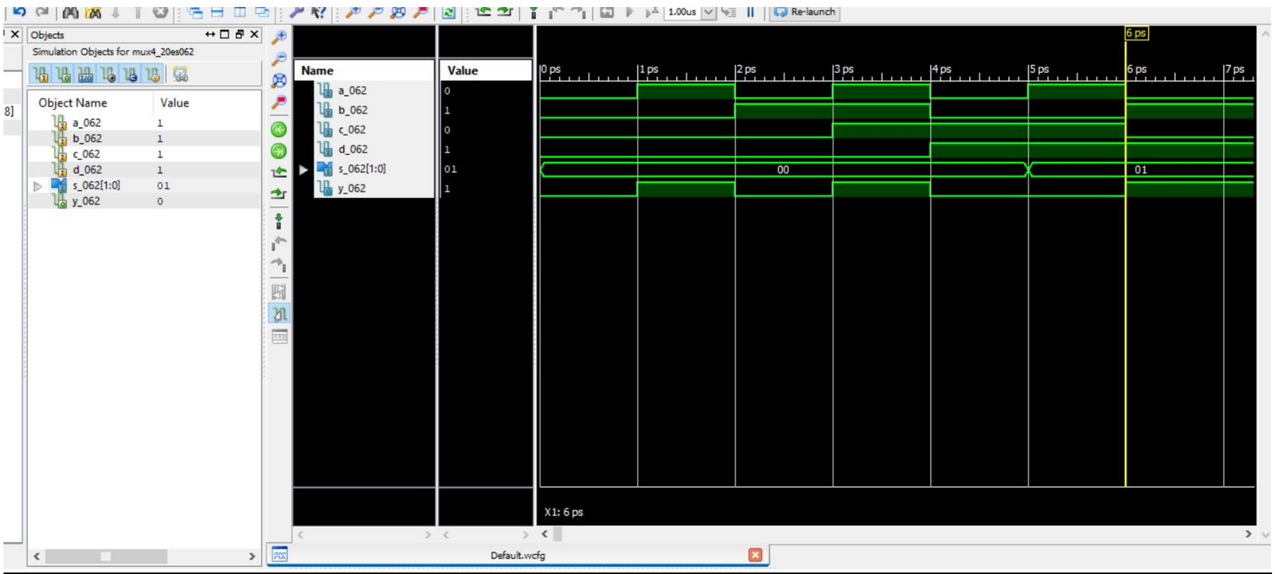


Schematic





Simulation



Object Name	Value
a_062	1
b_062	1
c_062	1
d_062	1
s_062[1:0]	01
y_062	0

Name	Value
a_062	0
b_062	1
c_062	0
d_062	1
s_062[1:0]	01
y_062	1