Spring 2022-ECGR-5146/4146-Introduction to VHDL
Dr. Fareena Saqib

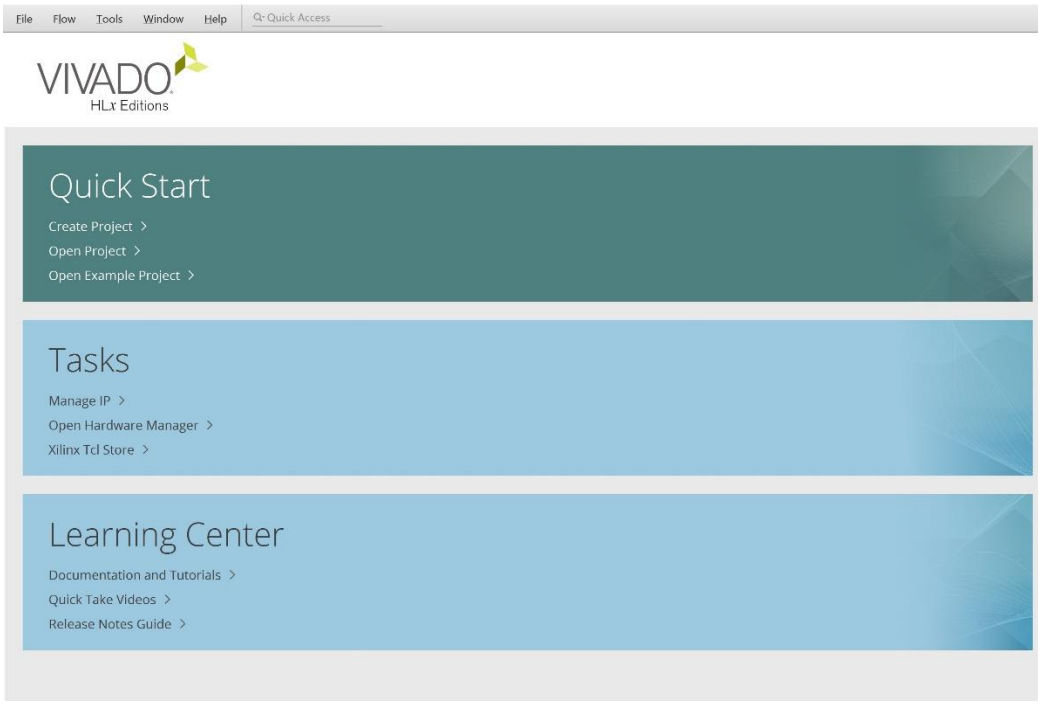
LAB 2

Full Adder

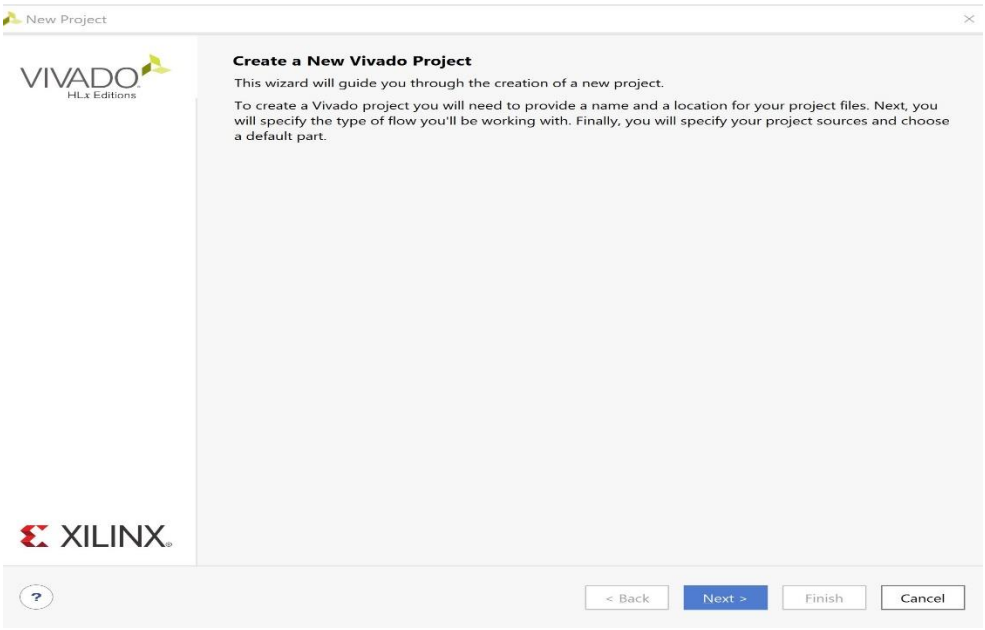
In this lab you will follow the step-by-step instructions provided and can also refer to the lecture slides **3-UNCC-VHDL-lecture_2_Introduction (slide no: 29-38)** to design a full adder.

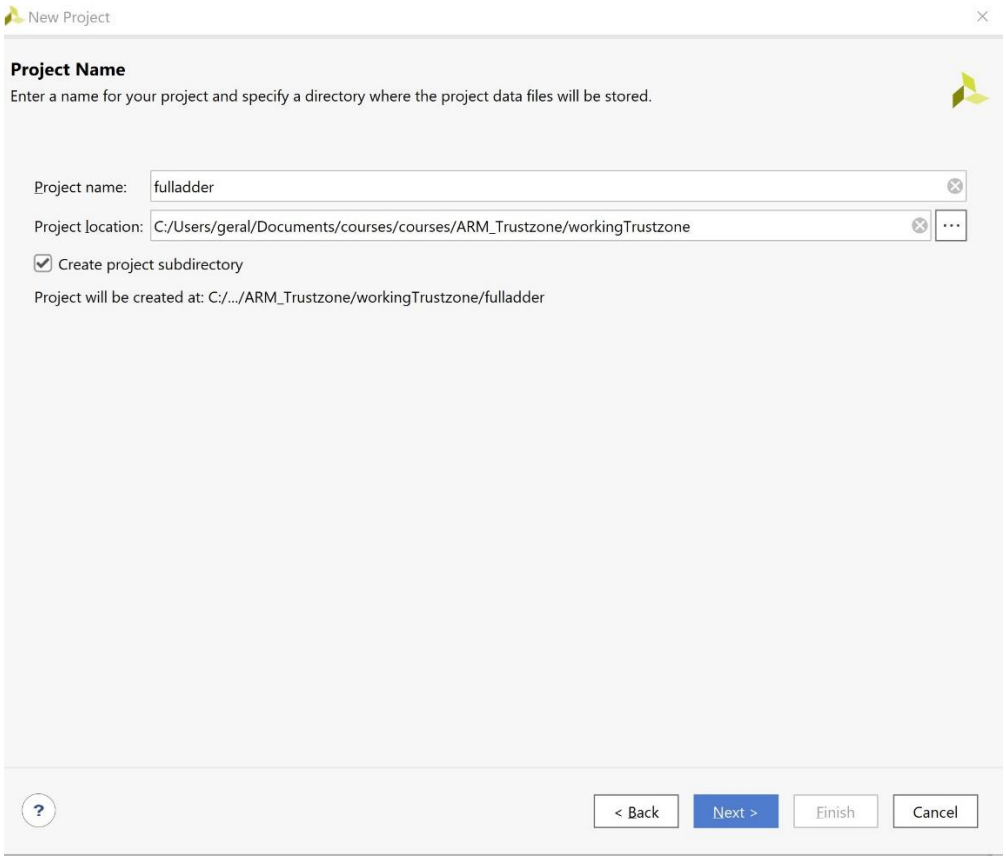
Step 1:

- On Windows, launch the Vivado IDE: Start > Vivado 2020.1
- In the Vivado IDE Getting started page, click **Create Project**

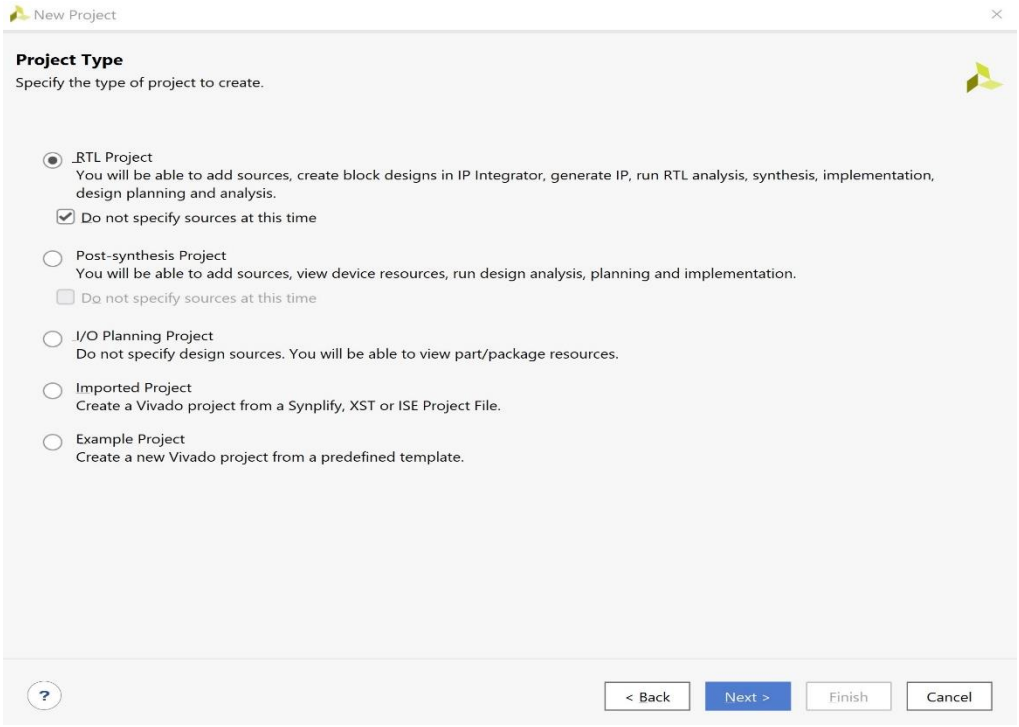


- In the **New project** dialog box, click Next, and enter a project name: ***fulladder***

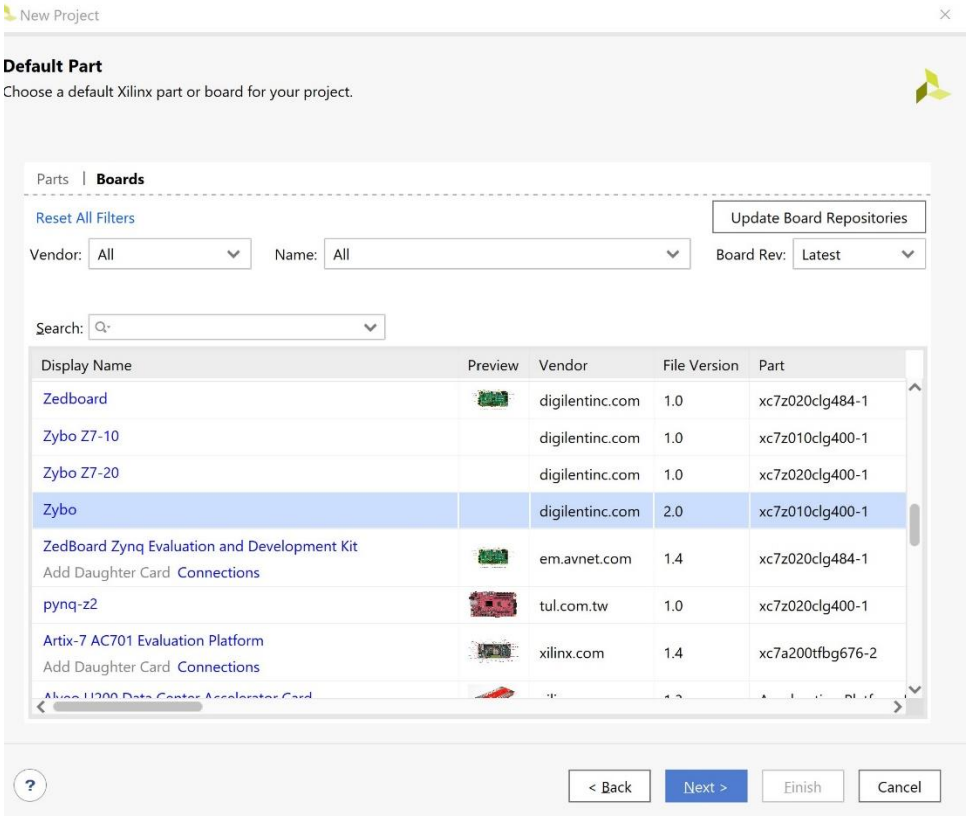




- In the Project Type dialog box, select RTL Project and check the do not specify sources at this time check box and click Next.



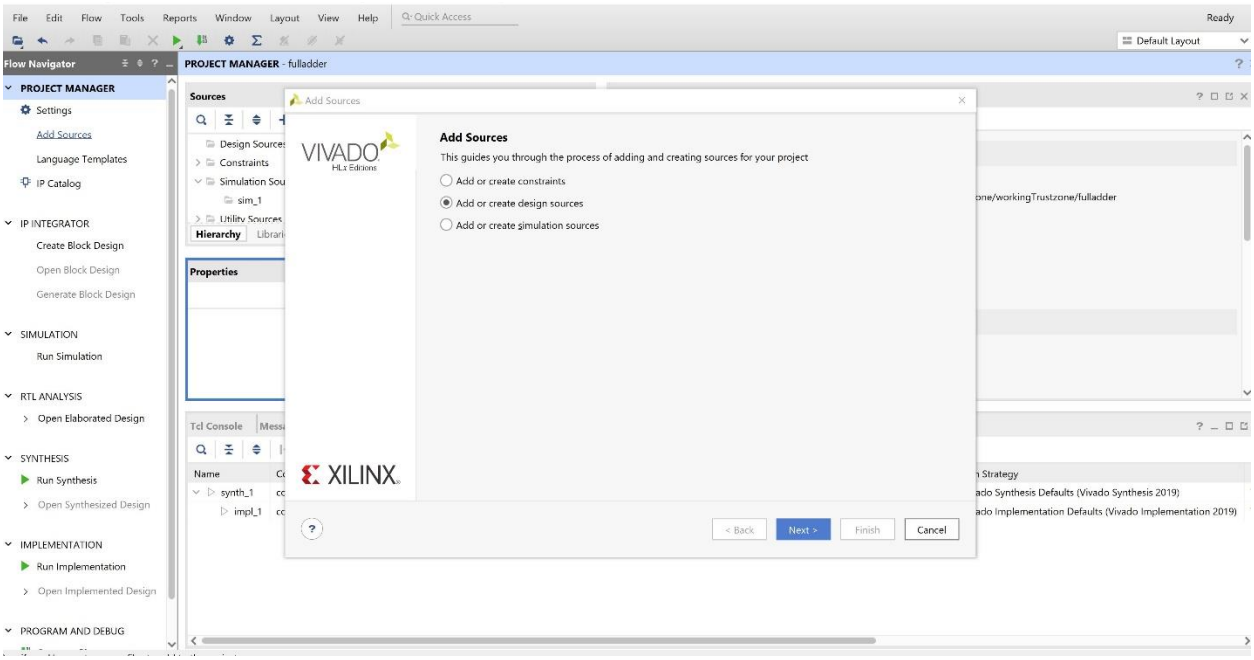
- In the Default Part dialog box select Boards, and then select either Zybo or Pynq board and click Next.



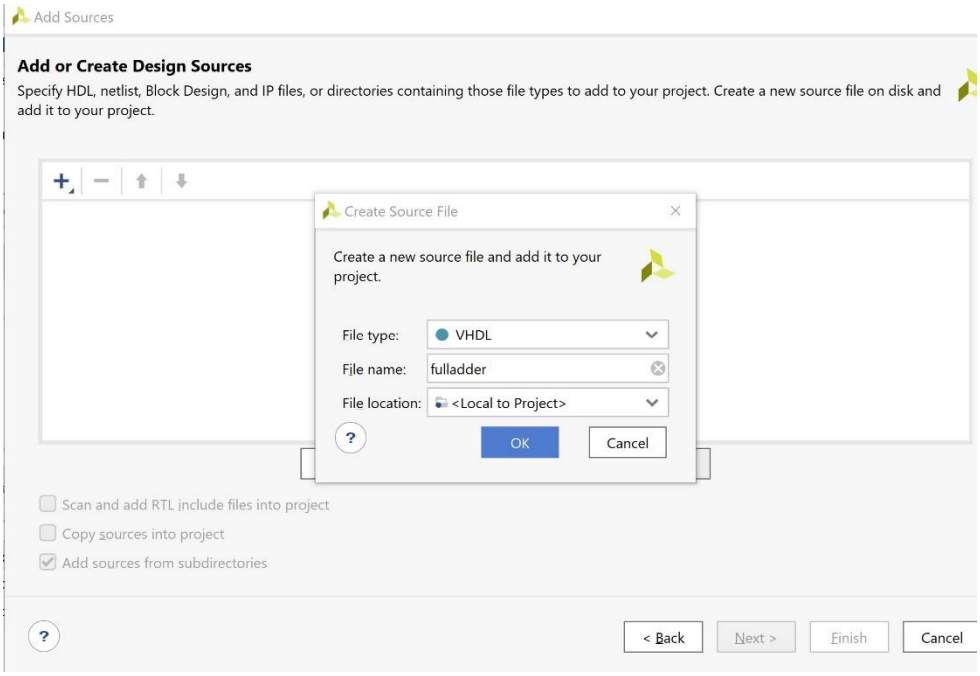
- Review the New Project Summary dialog box and click Finish to create the project.

Step 2:

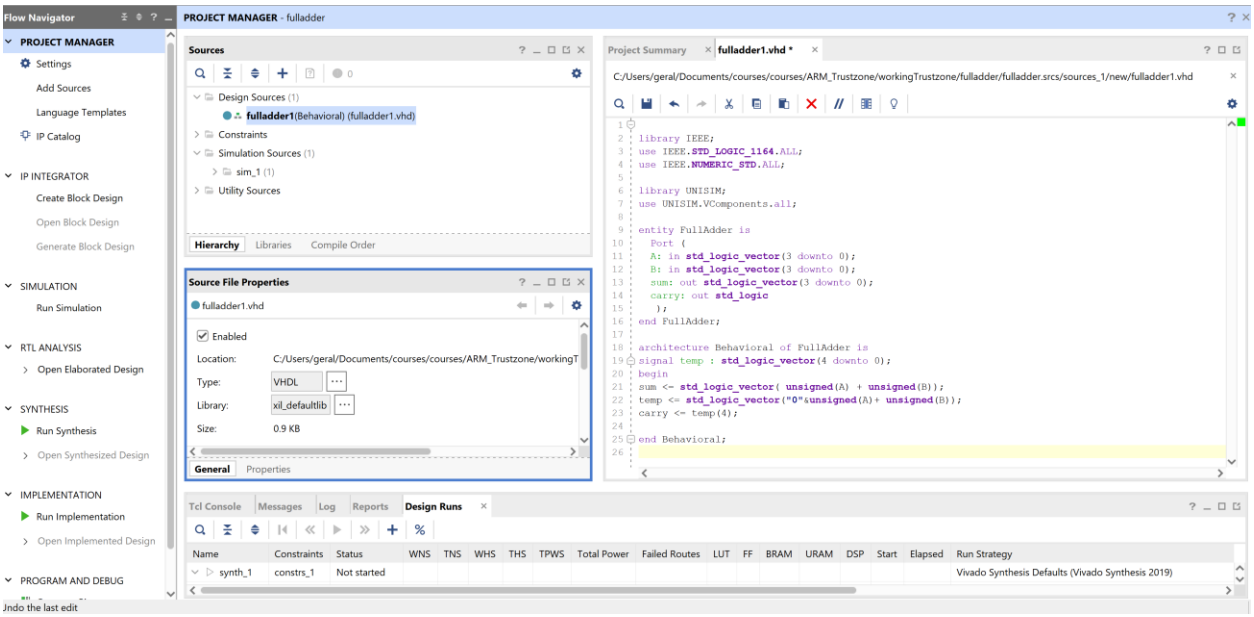
- In the project manager window click **add sources** and select **add or create design sources** and select next.



- Select the **create file** option and choose **VHDL** for the file type, enter the file name: **fulladder** and click ok.



- The Sources window displays the design source files. Click on the fulladder.vhd file and enter the provided full adder code.



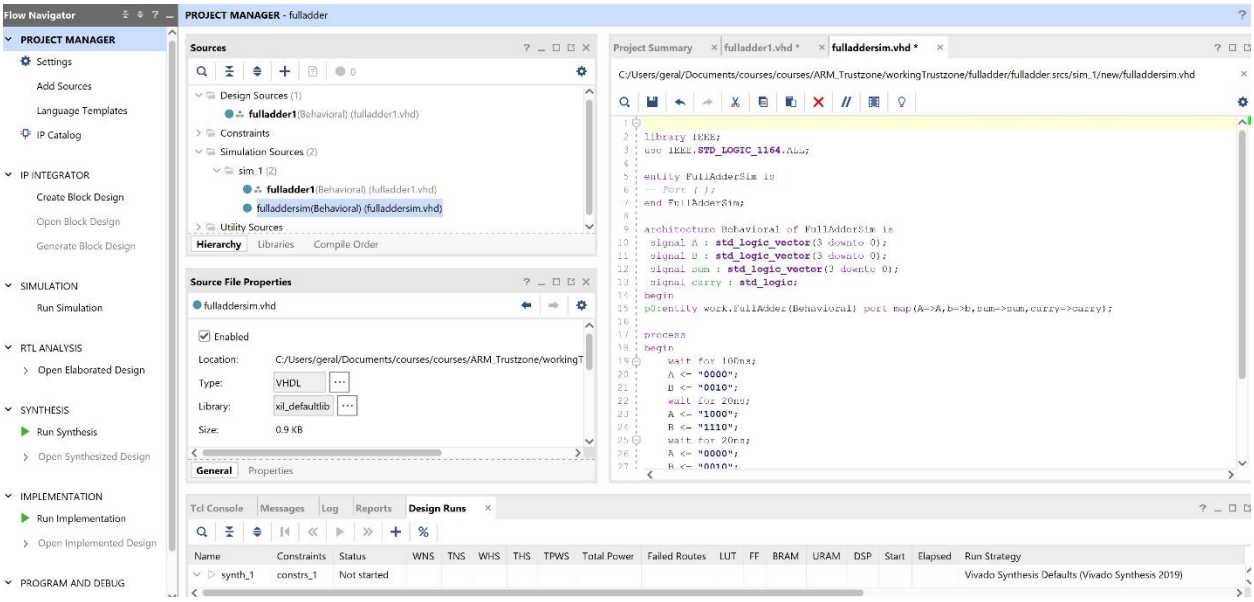
Fulladder.vhd code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;

entity FullAdder is
  Port (
  A: in std_logic_vector(3 downto 0);
  B: in std_logic_vector(3 downto 0);
  sum: out std_logic_vector(3 downto 0);
  carry: out std_logic
   );
end FullAdder;

architecture Behavioral of FullAdder is
signal temp : std_logic_vector(4 downto 0);
begin
sum <= std_logic_vector( unsigned(A) + unsigned(B));
temp <= std_logic_vector("0"&unsigned(A)+ unsigned(B));
carry <= temp(4);

end Behavioral;
```

- Similarly, in the project manager window click **add sources** and select **add or create simulation sources** and select next.

- Select the **create file** option and choose **VHDL** for the file type, enter the file name: **fulladdersim** and click ok.

- The Sources window displays the simulation source files. Click on the fulladdersim.vhd file and enter the provided full adder testbench code.



Fulladdersim.vhd Testbench:

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

entity FullAdderSim is
-- Port ( );
end FullAdderSim;

architecture Behavioral of FullAdderSim is
 signal A : std_logic_vector(3 downto 0);
 signal B : std_logic_vector(3 downto 0);
 signal sum : std_logic_vector(3 downto 0);
 signal carry : std_logic;
begin
p0:entity work.FullAdder(Behavioral) port map(A=>A,b=>b,sum=>sum,carry=>carry);

process
begin
    wait for 100ns;
    A <= "0000";
    B <= "0010";
    wait for 20ns;
    A <= "1000";
    B <= "1110";
    wait for 20ns;
```

```
    A <= "0000";
    B <= "0010";
    wait for 20ns;
    A <= "0000";
    B <= "0010";
    wait for 20ns;
  end process;
end Behavioral;
```

- Once the source file and simulation files are created run the synthesis and simulation process in the project manager to view the waveform for the full adder circuit.

Deliverables:

1. Take a screenshot of Vivado screen after you have run synthesis, and the simulation (Include the snapshot of the simulation waveform).

2. Once done with the simulation you need to make some changes in the fulladder.vhd code to generate the bitstream.

Change the highlighted area:

```
entity FullAdder is
    Port ( A : in STD_LOGIC_vector(1 downto 0);
           B : in STD_LOGIC_vector(1 downto 0);
           sum : out STD_LOGIC_vector(1 downto 0);
           carry : out STD_LOGIC);
end FullAdder;

architecture Behavioral of FullAdder is
signal temp : std_logic_vector(2 downto 0);
begin
sum <= std_logic_vector( unsigned(A) + unsigned(B));
temp <= std_logic_vector( "0" & unsigned(A) + unsigned(B));
carry <= temp(2);
```

3. Add the constraint file provided in the constraint tab of vivado.

4. Run synthesis, post place, and route and Generate Bitstream successfully. (1 snapshot needed after the bitstream)