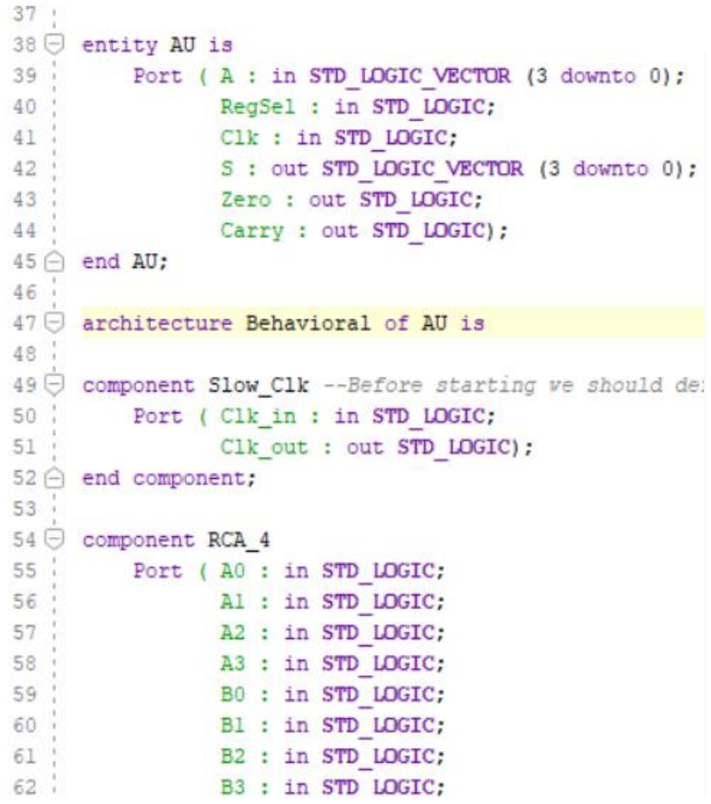
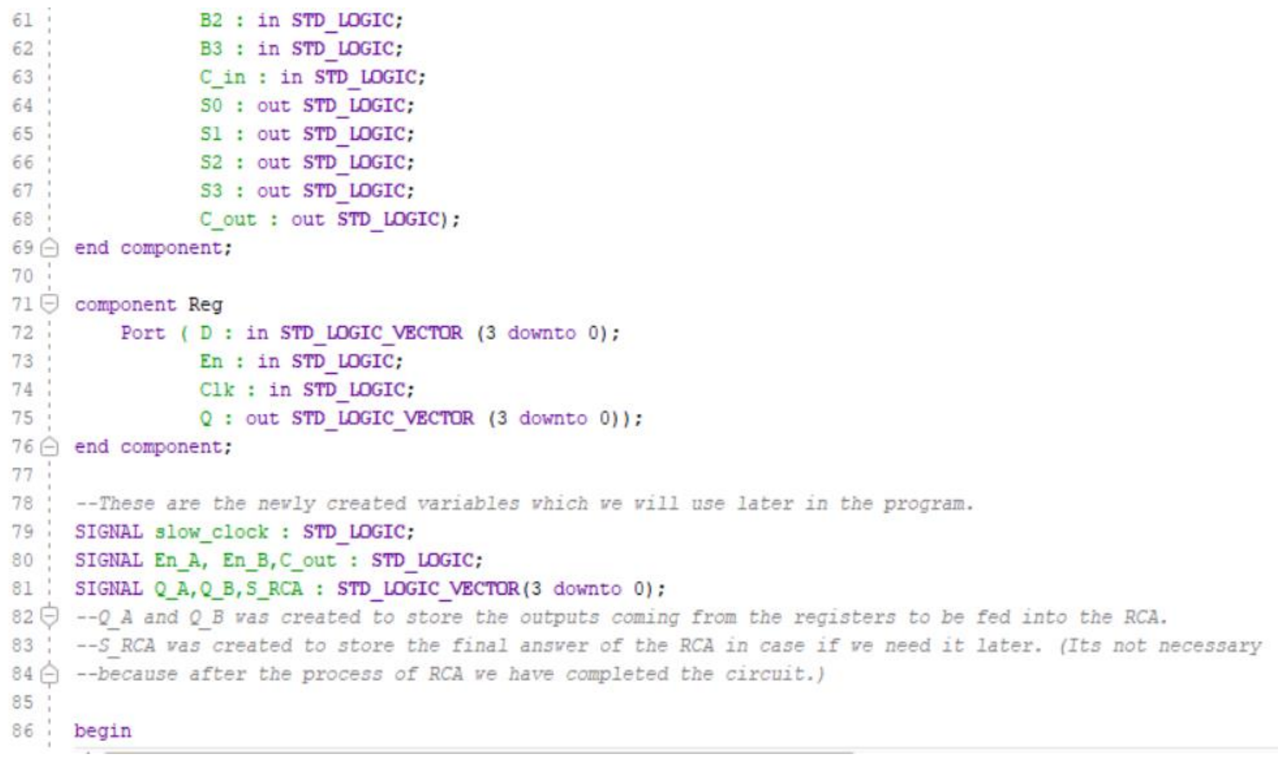
LAB 06 COMPUTER ORGANISATION & DIGITAL DESIGN

210495G Nipun Viraj

VHDL Codes (AU)

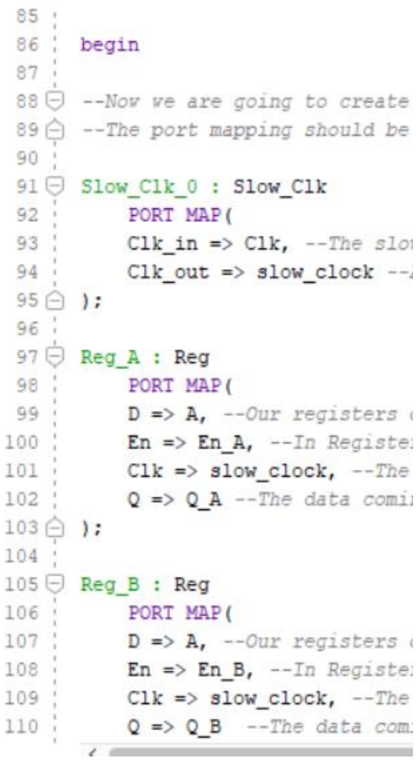


```

entity AU is
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           RegSel : in STD_LOGIC;
           Clk : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Zero : out STD_LOGIC;
           Carry : out STD_LOGIC);
end AU;

architecture Behavioral of AU is

component Slow_Clk --Before starting we should de
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end component;

component RCA_4
    Port ( A0 : in STD_LOGIC;
           A1 : in STD_LOGIC;
           A2 : in STD_LOGIC;
           A3 : in STD_LOGIC;
           B0 : in STD_LOGIC;
           B1 : in STD_LOGIC;
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
```

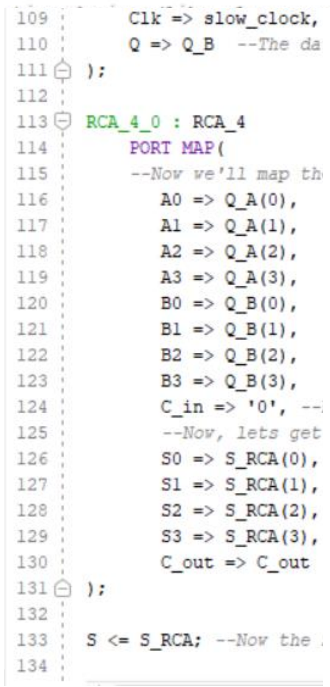


```
           B2 : in STD_LOGIC;
           B3 : in STD_LOGIC;
           C_in : in STD_LOGIC;
           S0 : out STD_LOGIC;
           S1 : out STD_LOGIC;
           S2 : out STD_LOGIC;
           S3 : out STD_LOGIC;
           C_out : out STD_LOGIC);
end component;

component Reg
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end component;

--These are the newly created variables which we will use later in the program.
SIGNAL slow_clock : STD_LOGIC;
SIGNAL En_A, En_B,C_out : STD_LOGIC;
SIGNAL Q_A,Q_B,S_RCA : STD_LOGIC_VECTOR(3 downto 0);
--Q_A and Q_B was created to store the outputs coming from the registers to be fed into the RCA.
--S_RCA was created to store the final answer of the RCA in case if we need it later. (Its not necessary
--because after the process of RCA we have completed the circuit.)

begin
```

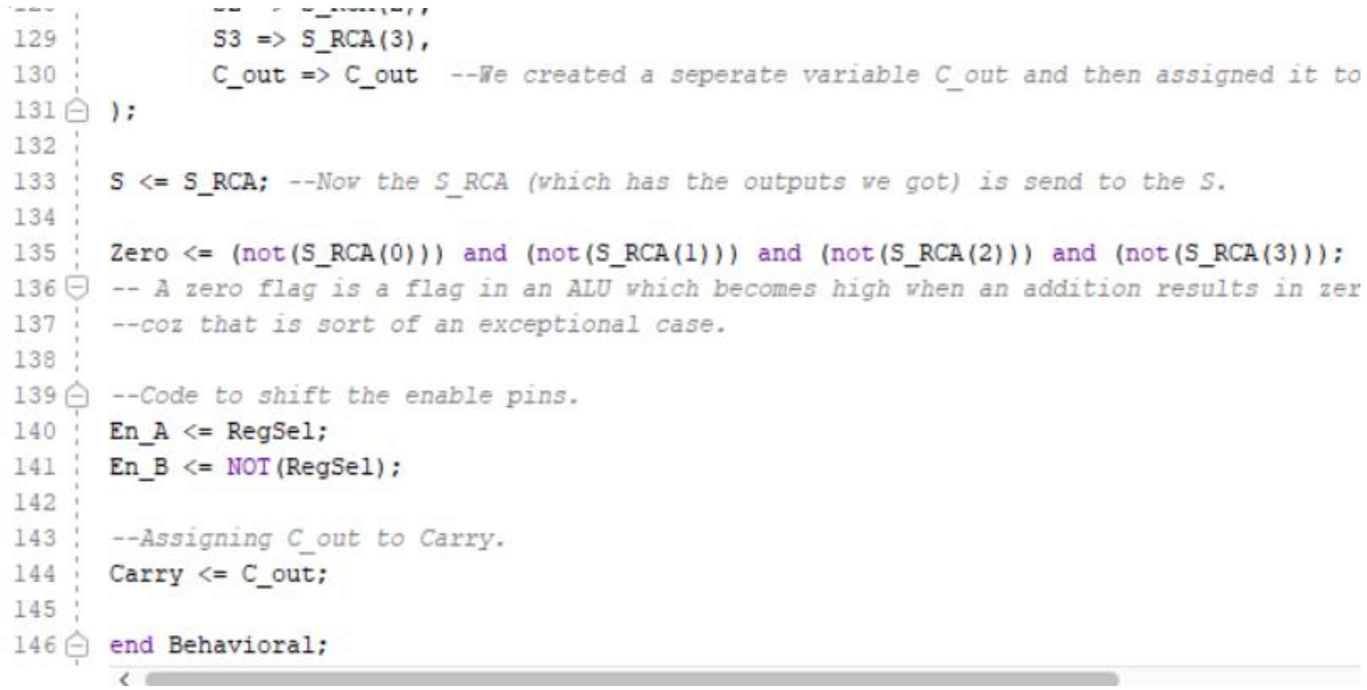
VHDL Codes (AU)



```
begin

--Now we are going to create
--The port mapping should be

Slow_Clk_0 : Slow_Clk
    PORT MAP(
    Clk_in => Clk, --The slo
    Clk_out => slow_clock --
);

Reg_A : Reg
    PORT MAP(
    D => A, --Our registers
    En => En_A, --In Registe
    Clk => slow_clock, --The
    Q => Q_A --The data comi
);

Reg_B : Reg
    PORT MAP(
    D => A, --Our registers
    En => En_B, --In Registe
    Clk => slow_clock, --The
    Q => Q_B  --The data com
```

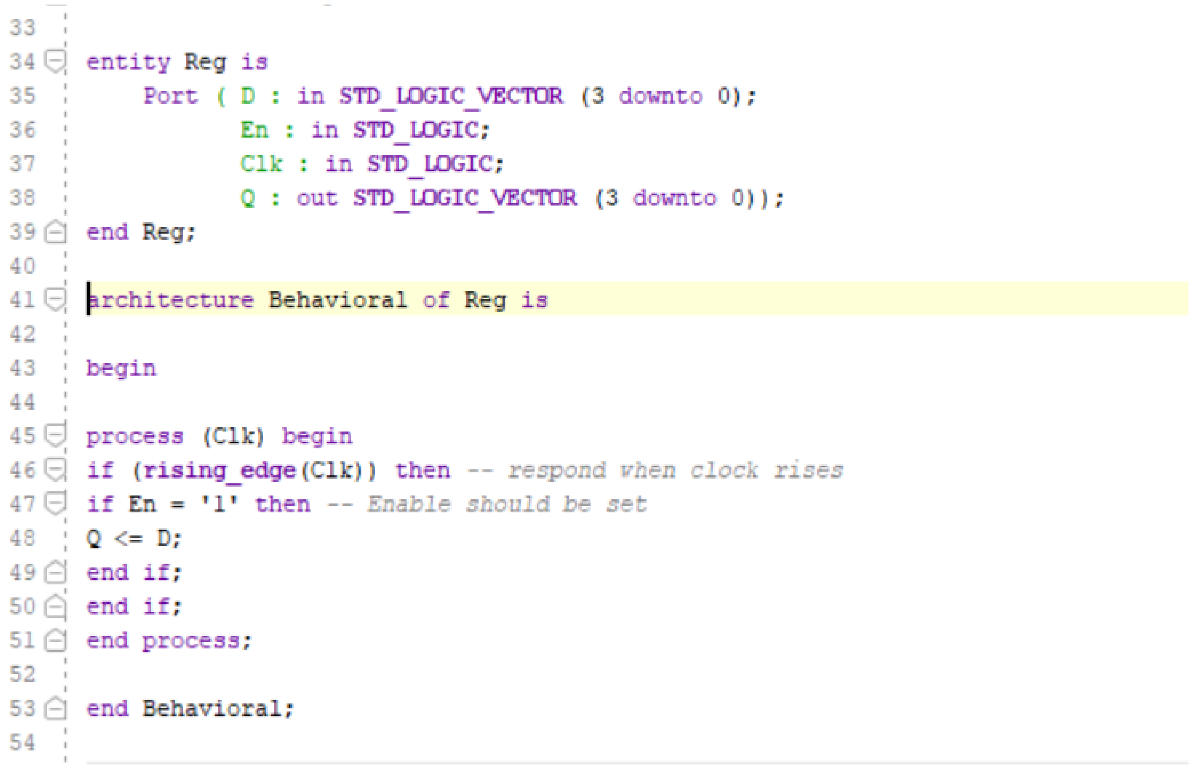


```
    Clk => slow_clock,
    Q => Q_B  --The da
);

RCA_4_0 : RCA_4
    PORT MAP(
    --Now we'll map th
       A0 => Q_A(0),
       A1 => Q_A(1),
       A2 => Q_A(2),
       A3 => Q_A(3),
       B0 => Q_B(0),
       B1 => Q_B(1),
       B2 => Q_B(2),
       B3 => Q_B(3),
       C_in => '0', --
       --Now, lets get
       S0 => S_RCA(0),
       S1 => S_RCA(1),
       S2 => S_RCA(2),
       S3 => S_RCA(3),
       C_out => C_out
);

S <= S_RCA; --Now the
```



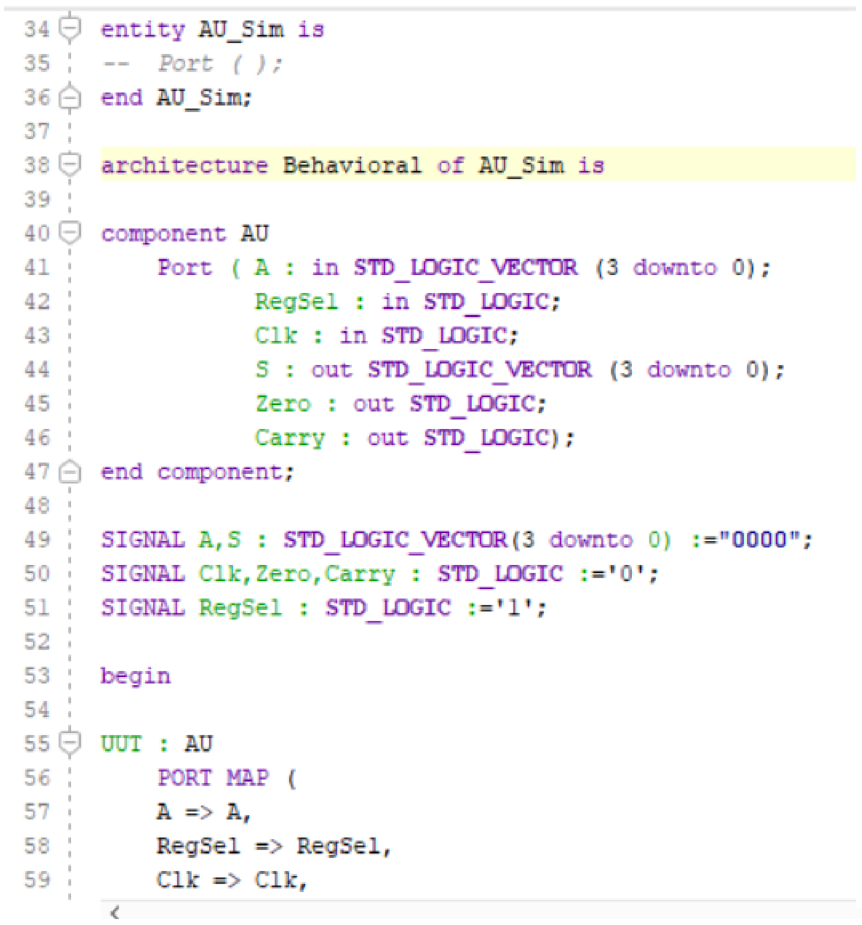
```
       S3 => S_RCA(3),
       C_out => C_out  --We created a seperate variable C_out and then assigned it to
);

S <= S_RCA; --Now the S_RCA (which has the outputs we got) is send to the S.

Zero <= (not(S_RCA(0))) and (not(S_RCA(1))) and (not(S_RCA(2))) and (not(S_RCA(3)));
-- A zero flag is a flag in an ALU which becomes high when an addition results in zer
--coz that is sort of an exceptional case.

--Code to shift the enable pins.
En_A <= RegSel;
En_B <= NOT(RegSel);

--Assigning C_out to Carry.
Carry <= C_out;

end Behavioral;
```

VHDL (Reg)

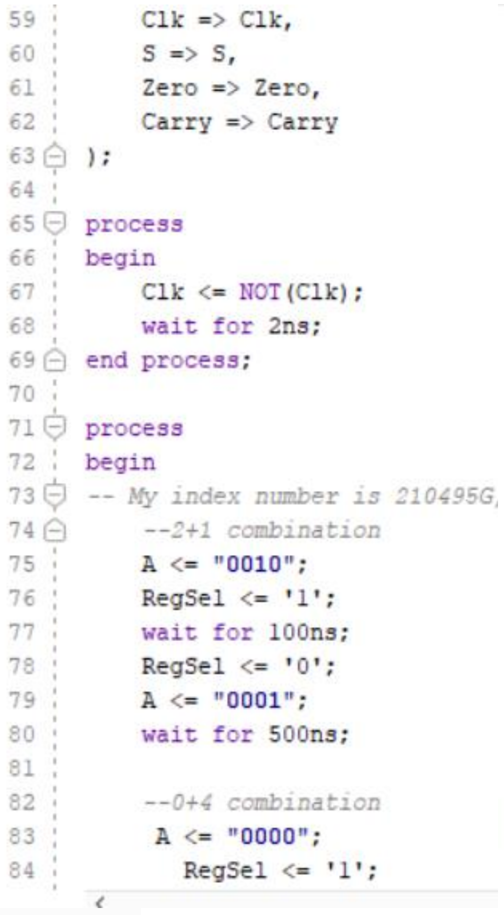


```
entity Reg is
    Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
           En : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end Reg;

architecture Behavioral of Reg is

begin

process (Clk) begin
if (rising_edge(Clk)) then -- respond when clock rises
if En = '1' then -- Enable should be set
Q <= D;
end if;
end if;
end process;

end Behavioral;
```

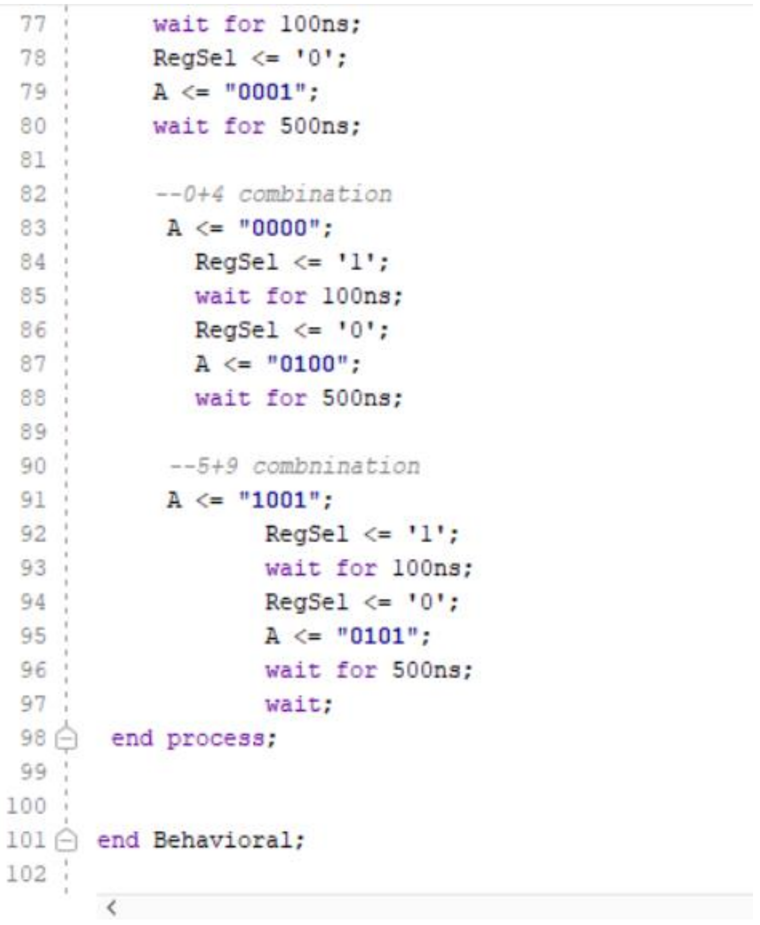
TB File AU



```
entity AU_Sim is
--  Port ( );
end AU_Sim;

architecture Behavioral of AU_Sim is

component AU
    Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
           RegSel : in STD_LOGIC;
           Clk : in STD_LOGIC;
           S : out STD_LOGIC_VECTOR (3 downto 0);
           Zero : out STD_LOGIC;
           Carry : out STD_LOGIC);
end component;

SIGNAL A,S : STD_LOGIC_VECTOR(3 downto 0) :="0000";
SIGNAL Clk,Zero,Carry : STD_LOGIC :='0';
SIGNAL RegSel : STD_LOGIC :='1';

begin

UUT : AU
    PORT MAP (
    A => A,
    RegSel => RegSel,
    Clk => Clk,
```

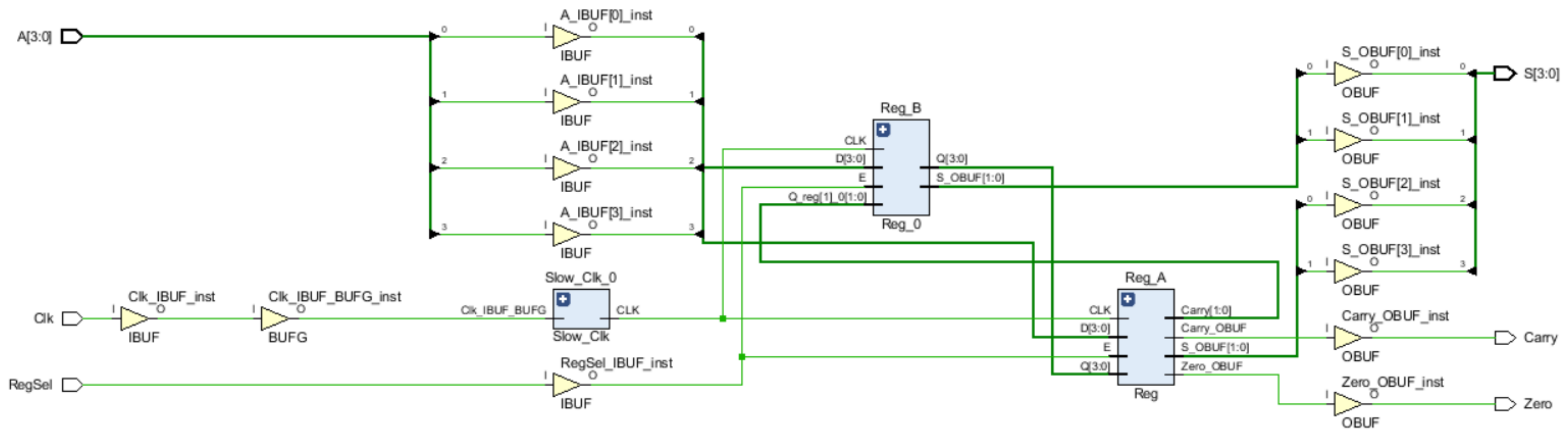


```
    Clk => Clk,
    S => S,
    Zero => Zero,
    Carry => Carry
);

process
begin
    Clk <= NOT(Clk);
    wait for 2ns;
end process;

process
begin
-- My index number is 210495G,
    --2+1 combination
    A <= "0010";
    RegSel <= '1';
    wait for 100ns;
    RegSel <= '0';
    A <= "0001";
    wait for 500ns;

    --0+4 combination
     A <= "0000";
       RegSel <= '1';
```



```
    wait for 100ns;
    RegSel <= '0';
    A <= "0001";
    wait for 500ns;

    --0+4 combination
     A <= "0000";
       RegSel <= '1';
       wait for 100ns;
       RegSel <= '0';
       A <= "0100";
       wait for 500ns;

     --5+9 combnination
     A <= "1001";
            RegSel <= '1';
            wait for 100ns;
            RegSel <= '0';
            A <= "0101";
            wait for 500ns;
            wait;
 end process;


end Behavioral;
```

AU Schematic



Conclusion

- In this lab, we use two registers to add two numbers.
- There is a register select which sends a signal to select the two registers at a time to feed in the values to the ripple carry adder.
- Then the numbers are added and the output is produces.