

LAB 06 COMPUTER ORGANISATION & DIGITAL DESIGN

210495G Nipun Viraj

VHDL Codes (AU)

```
37 :
38 entity AU is
39     Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
40           RegSel : in STD_LOGIC;
41           Clk : in STD_LOGIC;
42           S : out STD_LOGIC_VECTOR (3 downto 0);
43           Zero : out STD_LOGIC;
44           Carry : out STD_LOGIC);
45 end AU;
46
47 architecture Behavioral of AU is
48
49 component Slow_Clk --Before starting we should de:
50     Port ( Clk_in : in STD_LOGIC;
51           Clk_out : out STD_LOGIC);
52 end component;
53
54 component RCA_4
55     Port ( A0 : in STD_LOGIC;
56           A1 : in STD_LOGIC;
57           A2 : in STD_LOGIC;
58           A3 : in STD_LOGIC;
59           B0 : in STD_LOGIC;
60           B1 : in STD_LOGIC;
61           B2 : in STD_LOGIC;
62           B3 : in STD_LOGIC;
```

```
61           B2 : in STD_LOGIC;
62           B3 : in STD_LOGIC;
63           C_in : in STD_LOGIC;
64           S0 : out STD_LOGIC;
65           S1 : out STD_LOGIC;
66           S2 : out STD_LOGIC;
67           S3 : out STD_LOGIC;
68           C_out : out STD_LOGIC);
69 end component;
70
71 component Reg
72     Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
73           En : in STD_LOGIC;
74           Clk : in STD_LOGIC;
75           Q : out STD_LOGIC_VECTOR (3 downto 0));
76 end component;
77
78 --These are the newly created variables which we will use later in the program.
79 SIGNAL slow_clock : STD_LOGIC;
80 SIGNAL En_A, En_B, C_out : STD_LOGIC;
81 SIGNAL Q_A, Q_B, S_RCA : STD_LOGIC_VECTOR(3 downto 0);
82 --Q_A and Q_B was created to store the outputs coming from the registers to be fed into the RCA.
83 --S_RCA was created to store the final answer of the RCA in case if we need it later. (Its not necessary
84 --because after the process of RCA we have completed the circuit.)
85
86 begin
```

VHDL Codes (AU)

```

85
86 begin
87
88 --Now we are going to create
89 --The port mapping should be
90
91 Slow_Clk_0 : Slow_Clk
92   PORT MAP(
93     Clk_in => Clk, --The slow clock
94     Clk_out => slow_clock --The fast clock
95   );
96
97 Reg_A : Reg
98   PORT MAP(
99     D => A, --Our registers data
100     En => En_A, --In Register enable
101     Clk => slow_clock, --The clock
102     Q => Q_A --The data coming out
103   );
104
105 Reg_B : Reg
106   PORT MAP(
107     D => A, --Our registers data
108     En => En_B, --In Register enable
109     Clk => slow_clock, --The clock
110     Q => Q_B --The data coming out

```

```

109   Clk => slow_clock,
110   Q => Q_B --The data coming out
111 );
112
113 RCA_4_0 : RCA_4
114   PORT MAP(
115     --Now we'll map the inputs
116     A0 => Q_A(0),
117     A1 => Q_A(1),
118     A2 => Q_A(2),
119     A3 => Q_A(3),
120     B0 => Q_B(0),
121     B1 => Q_B(1),
122     B2 => Q_B(2),
123     B3 => Q_B(3),
124     C_in => '0', --Initial carry
125     --Now, let's get the outputs
126     S0 => S_RCA(0),
127     S1 => S_RCA(1),
128     S2 => S_RCA(2),
129     S3 => S_RCA(3),
130     C_out => C_out
131 );
132
133 S <= S_RCA; --Now the S_RCA (which has the outputs we got) is sent to the S.
134

```

```

129   S3 => S_RCA(3),
130   C_out => C_out --We created a separate variable C_out and then assigned it to
131 );
132
133 S <= S_RCA; --Now the S_RCA (which has the outputs we got) is sent to the S.
134
135 Zero <= (not(S_RCA(0))) and (not(S_RCA(1))) and (not(S_RCA(2))) and (not(S_RCA(3)));
136 -- A zero flag is a flag in an ALU which becomes high when an addition results in zero
137 -- coz that is sort of an exceptional case.
138
139 --Code to shift the enable pins.
140 En_A <= RegSel;
141 En_B <= NOT(RegSel);
142
143 --Assigning C_out to Carry.
144 Carry <= C_out;
145
146 end Behavioral;

```

VHDL (Reg)

```
33
34 entity Reg is
35     Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
36           En : in STD_LOGIC;
37           Clk : in STD_LOGIC;
38           Q : out STD_LOGIC_VECTOR (3 downto 0));
39 end Reg;
40
41 architecture Behavioral of Reg is
42
43     begin
44
45     process (Clk) begin
46         if (rising_edge(Clk)) then -- respond when clock rises
47             if En = '1' then -- Enable should be set
48                 Q <= D;
49             end if;
50         end if;
51     end process;
52
53 end Behavioral;
54
```

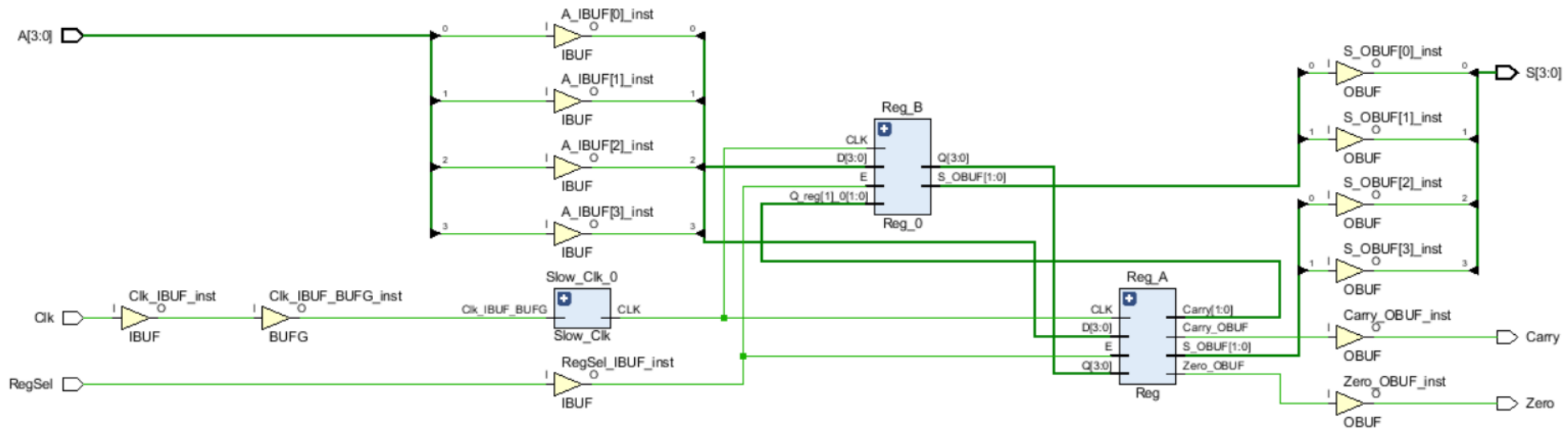
TB File AU

```
34 entity AU_Sim is
35   -- Port ( );
36 end AU_Sim;
37
38 architecture Behavioral of AU_Sim is
39
40   component AU
41     Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
42           RegSel : in STD_LOGIC;
43           Clk : in STD_LOGIC;
44           S : out STD_LOGIC_VECTOR (3 downto 0);
45           Zero : out STD_LOGIC;
46           Carry : out STD_LOGIC);
47   end component;
48
49   SIGNAL A,S : STD_LOGIC_VECTOR(3 downto 0) := "0000";
50   SIGNAL Clk,Zero,Carry : STD_LOGIC := '0';
51   SIGNAL RegSel : STD_LOGIC := '1';
52
53   begin
54
55   UUT : AU
56     PORT MAP (
57       A => A,
58       RegSel => RegSel,
59       Clk => Clk,
```

```
59       Clk => Clk,
60       S => S,
61       Zero => Zero,
62       Carry => Carry
63   );
64
65   process
66   begin
67     Clk <= NOT(Clk);
68     wait for 2ns;
69   end process;
70
71   process
72   begin
73     -- My index number is 210495G,
74     --2+1 combination
75     A <= "0010";
76     RegSel <= '1';
77     wait for 100ns;
78     RegSel <= '0';
79     A <= "0001";
80     wait for 500ns;
81
82     --0+4 combination
83     A <= "0000";
84     RegSel <= '1';
```

```
77     wait for 100ns;
78     RegSel <= '0';
79     A <= "0001";
80     wait for 500ns;
81
82     --0+4 combination
83     A <= "0000";
84     RegSel <= '1';
85     wait for 100ns;
86     RegSel <= '0';
87     A <= "0100";
88     wait for 500ns;
89
90     --5+9 combination
91     A <= "1001";
92     RegSel <= '1';
93     wait for 100ns;
94     RegSel <= '0';
95     A <= "0101";
96     wait for 500ns;
97     wait;
98   end process;
99
100
101 end Behavioral;
102
```

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 produced.