DEVELOPING VHDL CODE AND RUNNING IT ON CPLD BOARD

Aim: Forming VHDL code and feeding it onto CPLD board and checking its functionality.

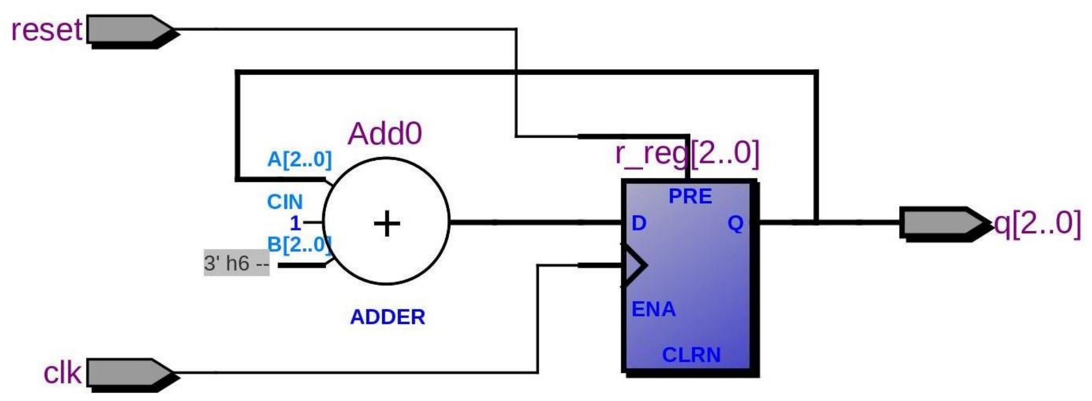
Summary of the experiment: Developing VHDL code for the following in behavioral style:

- 3-bit Up Counter
- 3-bit Down Counter.
- 3-bit Any Sequence Counter (3 → 0 → 2 → 5 → 1 → 4 → 3)
- 3-bit Ring counter
- 3-bit Johnson counter

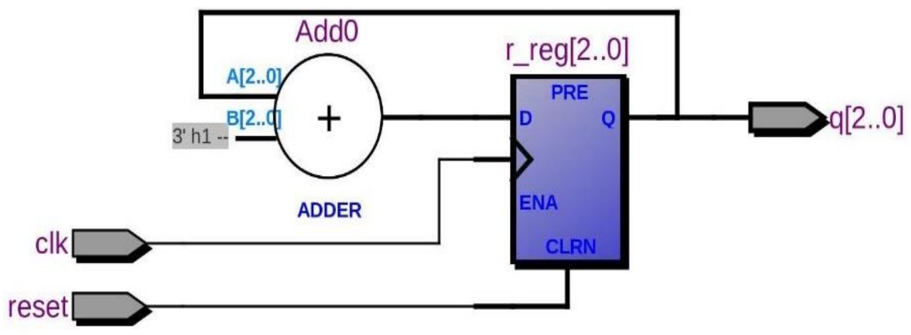
Components used: CPLD MAX3000A Board, JTAG port, Type-B USB cable, ALTERA BUS Blaster cable, Bus Blaster HDL code (for CPLD)

Snapshots of Gate-level Netlist:

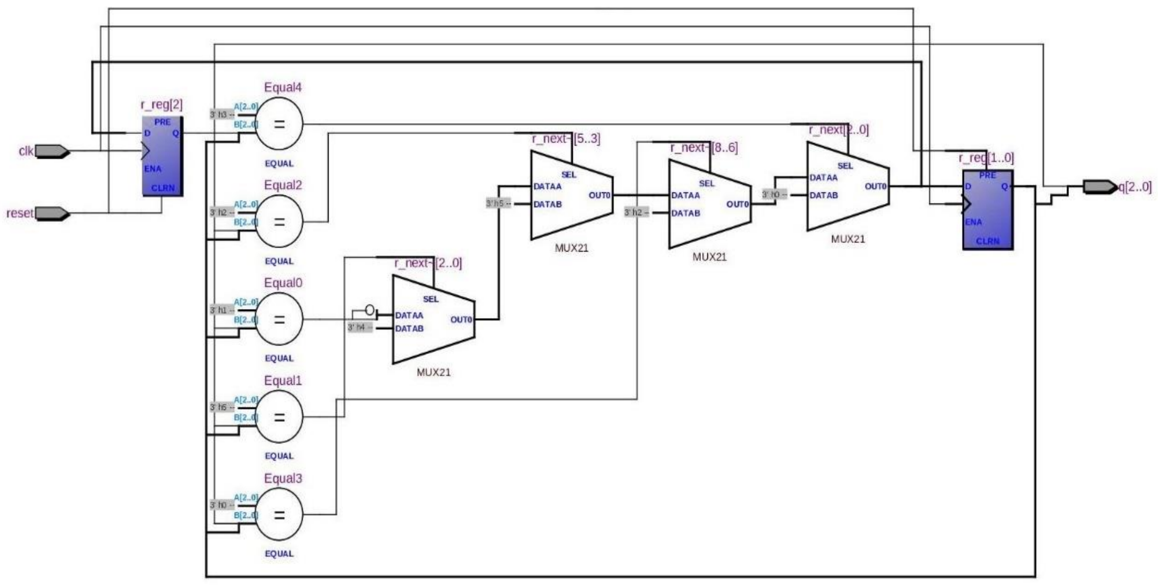
- ***3-bit Down Counter***



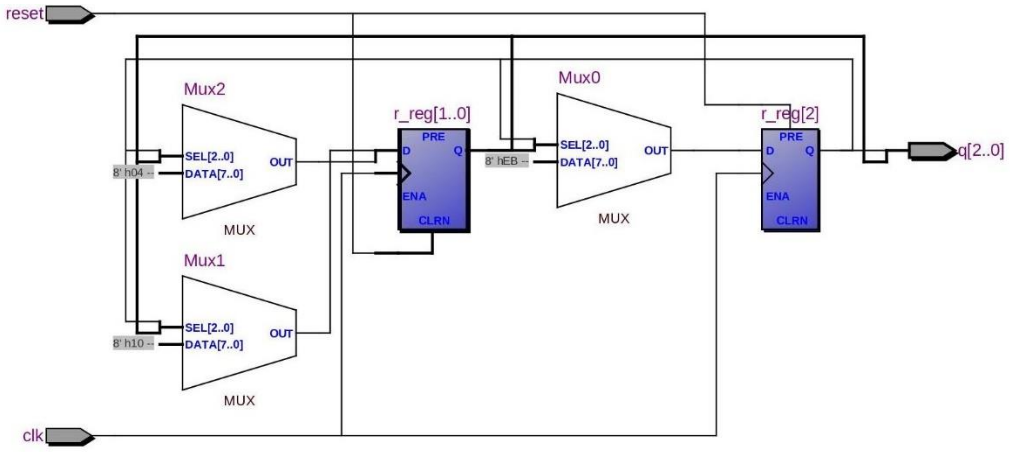
- ***3-bit Up Counter***



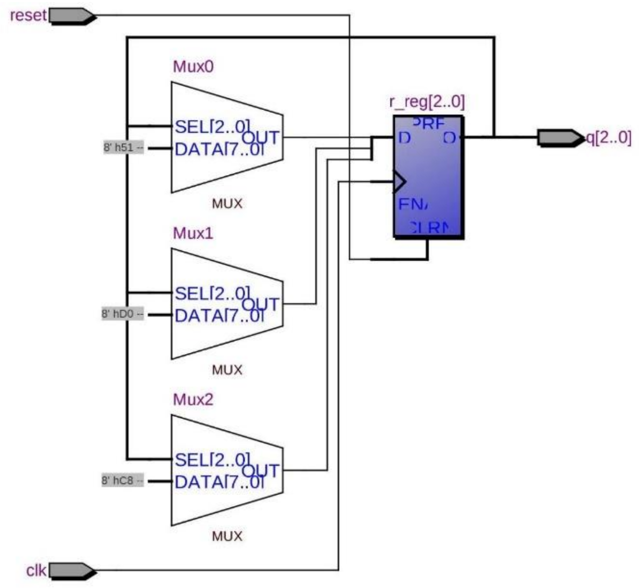
- ***3-bit Any Sequence Counter***



- ***3-bit Ring counter***

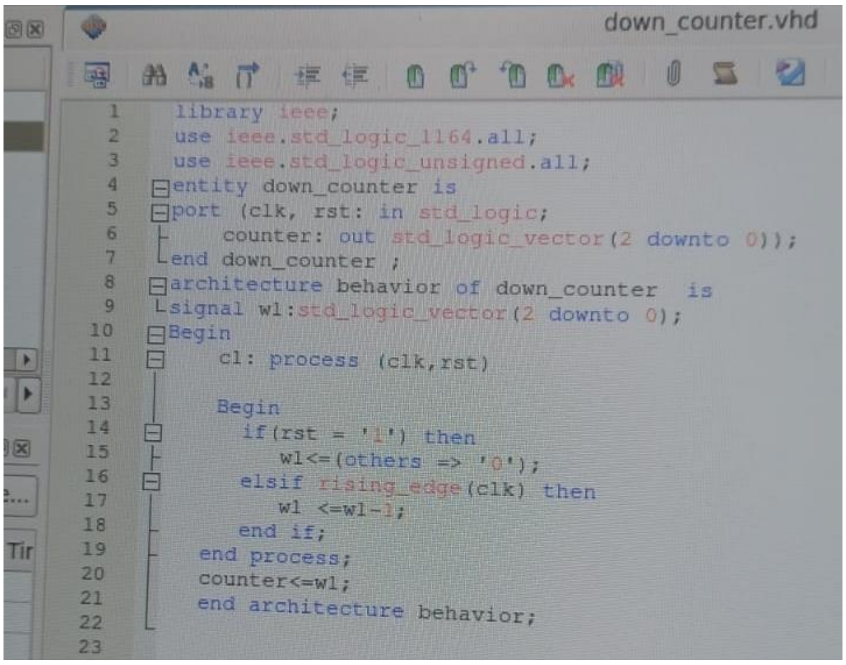


- ***3-bit Johnson counter***



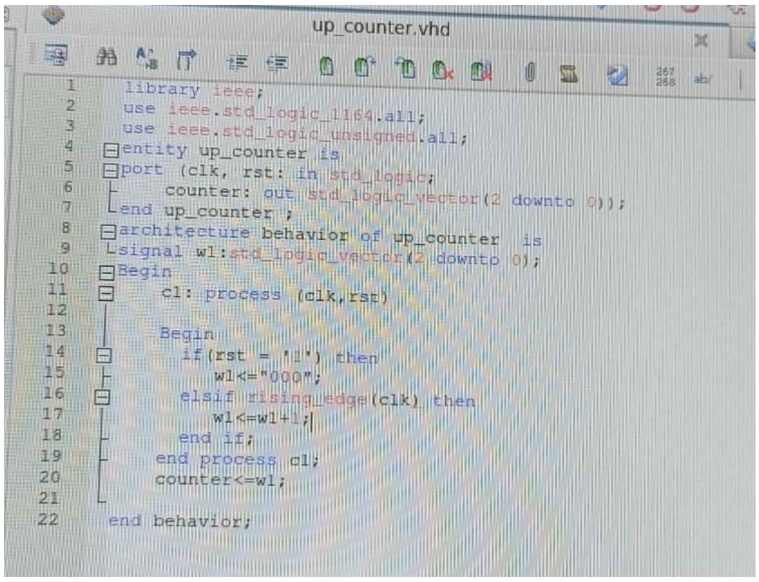
VHDL Code:

- ***3-bit Down Counter***



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity down_counter is
port (clk, rst: in std_logic;
     counter: out std_logic_vector(2 downto 0));
end down_counter ;
architecture behavior of down_counter  is
signal wl:std_logic_vector(2 downto 0);
Begin
    cl: process (clk,rst)

    Begin
      if(rst = '1') then
         wl<=(others => '0');
      elsif rising_edge(clk) then
         wl <=wl-1;
      end if;
  end process;
  counter<=wl;
  end architecture behavior;
```

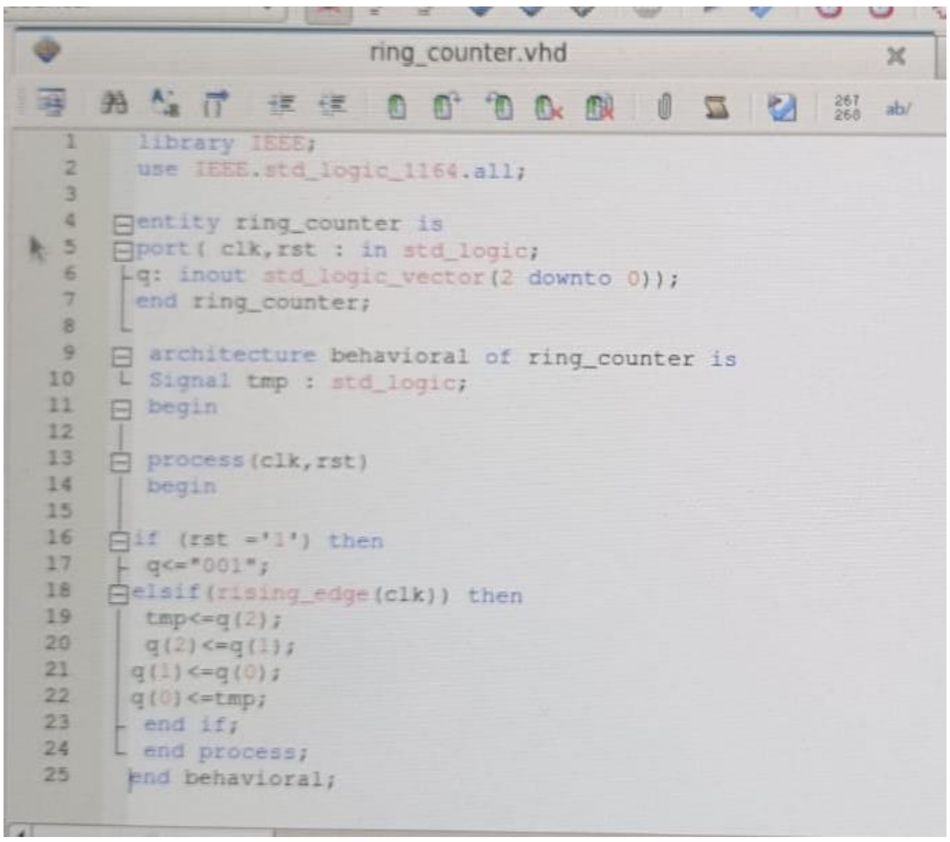
- ***3-bit Up Counter***



up_counter.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity up_counter is
port (clk, rst: in std_logic;
     counter: out std_logic_vector(2 downto 0));
end up_counter ;
architecture behavior of up_counter  is
signal w1:std_logic_vector(2 downto 0);
Begin
     c1: process (clk,rst)

     Begin
       if(rst = '1') then
           w1<="000";
       elsif rising_edge(clk) then
           w1<=w1+1;
       end if;
     end process c1;
     counter<=w1;

 end behavior;
```

- ***3-bit Ring counter***



ring_counter.vhd

```
  library IEEE;
  use IEEE.std_logic_1164.all;

entity ring_counter is
port( clk,rst : in std_logic;
 q: inout std_logic_vector(2 downto 0));
 end ring_counter;

  architecture behavioral of ring_counter is
  Signal tmp : std_logic;
  begin

  process(clk,rst)
  begin

if (rst ='1') then
 q<="001";
elsif(rising_edge(clk)) then
  tmp<=q(2);
  q(2)<=q(1);
 q(1)<=q(0);
 q(0)<=tmp;
  end if;
  end process;
 end behavioral;
```

- ***3-bit Any Sequence Counter***

```
library ieee;
use ieee.std_logic_1164.all;

entity arb_seq_counter3 is
  port (clk, reset: in std_logic;
    q: out std_logic_vector (2 downto 0));
  end arb_seq_counter3;

architecture two_seg_arch of arb_seq_counter3 is

signal r_reg: std_logic_vector (2 downto 0);
signal r_next: std_logic_vector (2 downto 0);
```

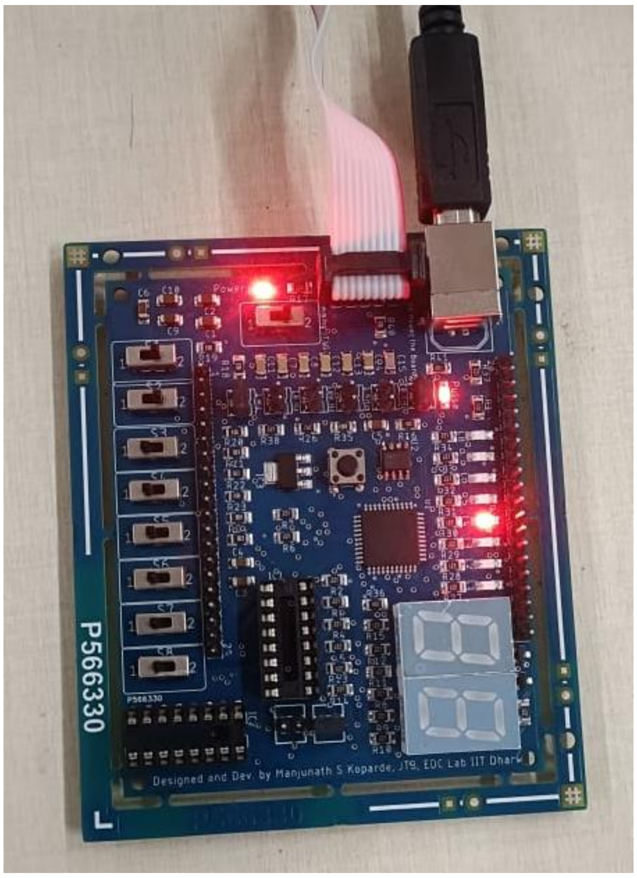
```
begin
process (clk, reset)
  begin
    if (reset='1') then
      r_reg <= "011";
      elsif (clk'event and clk='1') then
      r_reg <= r_next;
    end if;
  end process;
r_next <= "000" when r_reg="011"
else
"010" when r_reg="000"
else
"101" when r_reg="010"
else
"001" when r_reg="101"
else
"100" when r_reg="001"
else
"011";
q <= r_reg;
end two_seg_arch;
```

- ***3-bit Johnson Counter***

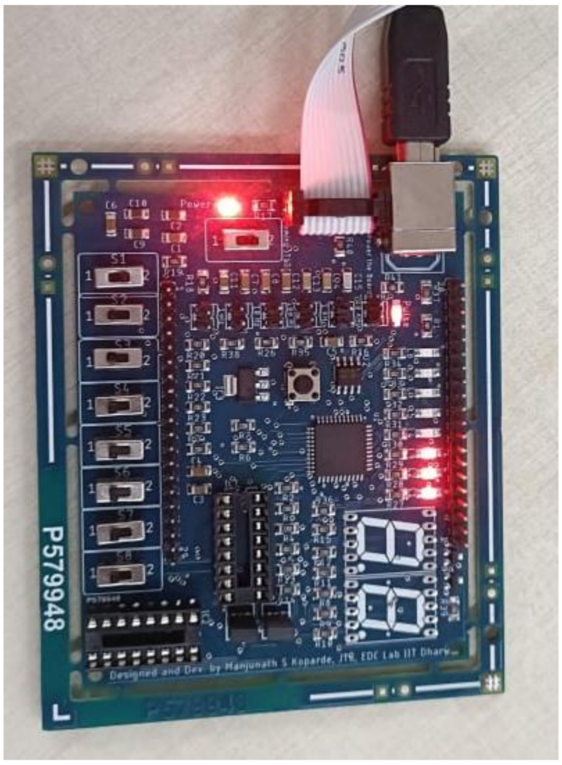
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity johnson_counter is
  port (clk, reset: in std_logic;
    q: out std_logic_vector (2 downto 0));
  end johnson_counter;

architecture two_seg_arch of johnson_counter is

signal r_reg: std_logic_vector (2 downto 0);
signal r_next: std_logic_vector (2 downto 0);

begin
process (clk, reset)
  begin
    if (reset='1') then
      r_reg <= "000";
      elsif (clk'event and clk='1') then
      r_reg <= r_next;
    end if;

    case r_reg is
      when "000" =>
        r_next <= "100";
      when "100" =>
        r_next <= "110";
      when "110" =>
        r_next <= "111";
      when "111" =>
        r_next <= "011";
      when "011" =>
        r_next <= "001";
      when "001" =>
        r_next <= "000";
      when "101" | "010" => -- all invalids states go to "000"
        r_next <= "000";
    end case;
    q <= r_reg;
  end process;
end two_seg_arch;
```

CPLD Board image:

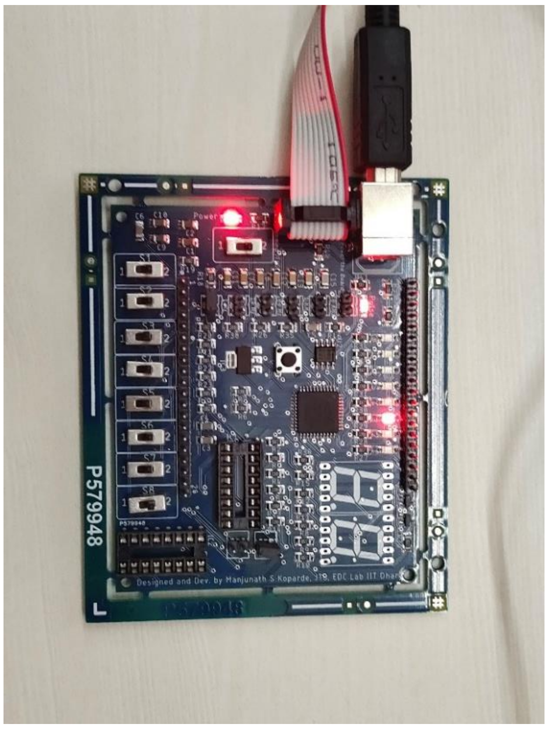
3-bit Up Counter


Power
P566330
Designed and Dev. by Manjunath S Koparde, JT9, EDC Lab IIT Dharw

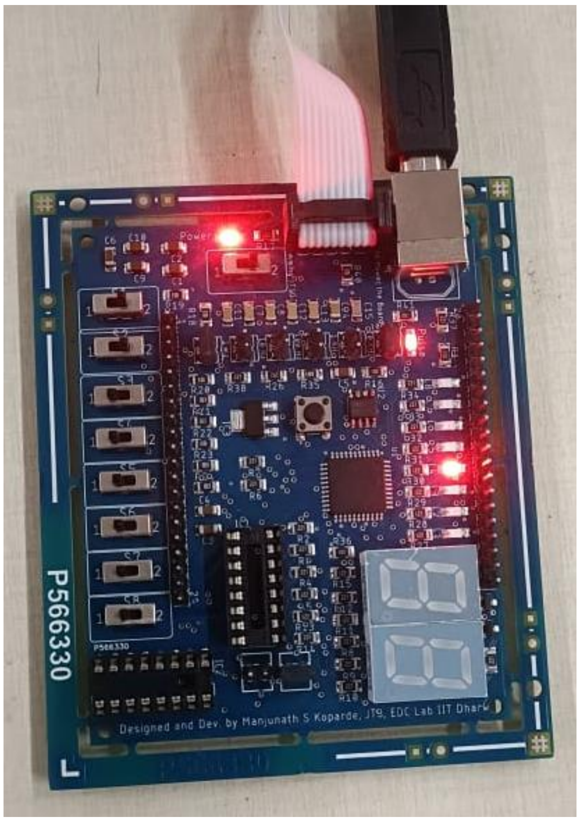
3-bit Down Counter


Power
P579948
Designed and Dev. by Manjunath S Koparde, JT9, EDC Lab IIT Dharw

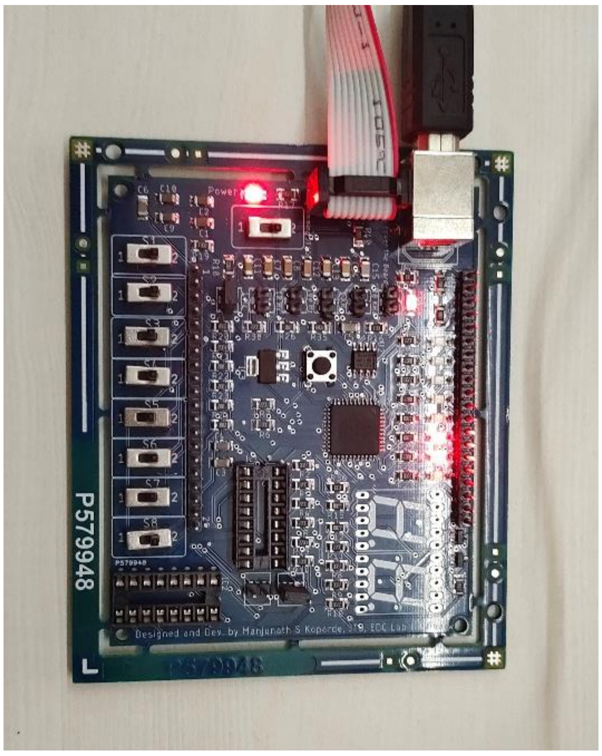
3-bit Any Sequence Counter


P579948
Designed and Dev. by Manjunath S Koparde, JT9, EDC Lab IIT Dharwad

3-bit ring counter


P566330
Designed and Dev. by Manjunath S Koparde, JT9, EDC Lab IIT Dharwad

3-bit Johnson counter



Results and Discussion: We discovered that because the circuits are asynchronous by design, their default state can be restored at any time, regardless of the clock. We found that, provided we know the order of the states, behavioral modeling allows us to efficiently write any counter, regardless of how complex it is.

Conclusion: We can infer from this experiment that hardware description languages are quite helpful when building complex circuits. Using VHDL code, we were able to quickly design a variety of counters for this experiment. These circuits could have been made, but employing logic gates would have been exceedingly difficult. So, we may conclude that hardware description languages are very beneficial for creating complex circuits fast in prototype form. is.