CPLD MAX3000A

Counters in VHDL

10th April, 2023

VHDL Programming and CPLD Application the VHDL codes.

Components used:

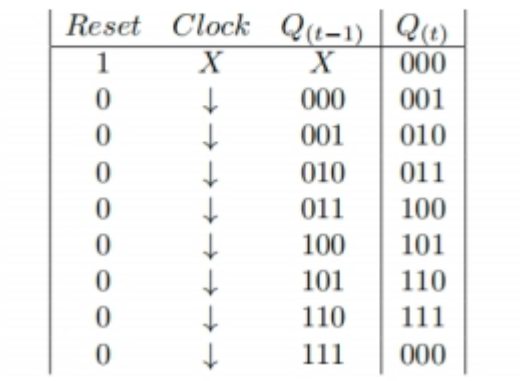
Quartus software,CPLD Board and USB connector

Summary of the Experiment:

Using Quartus application to write the VHDL code for the given problems and dump the code the CPLD(blue board).

Truth Table(s):

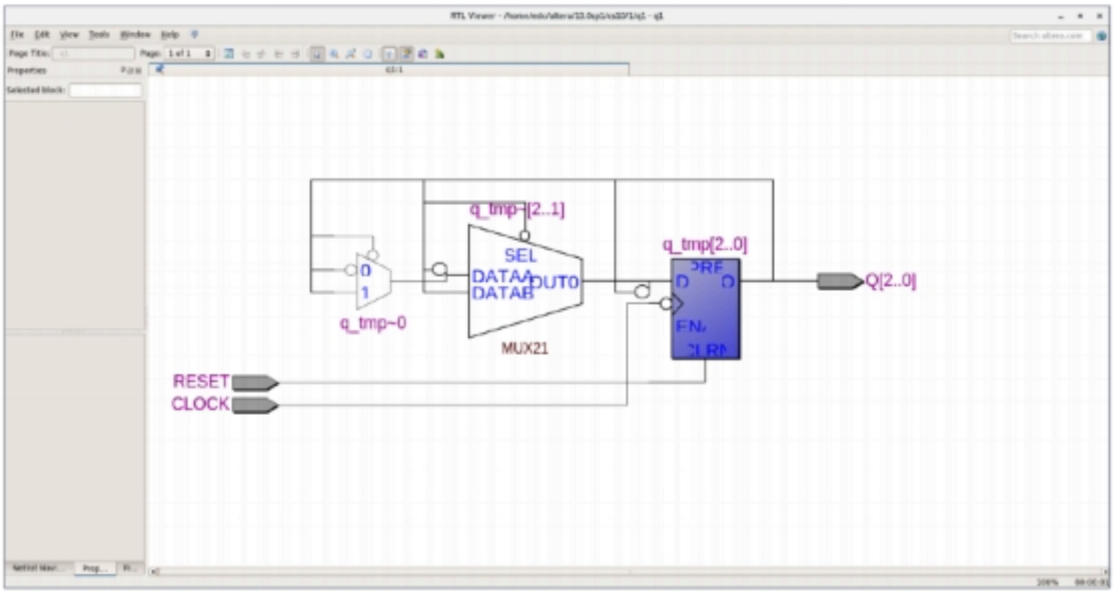
1. Asynchronous up counter

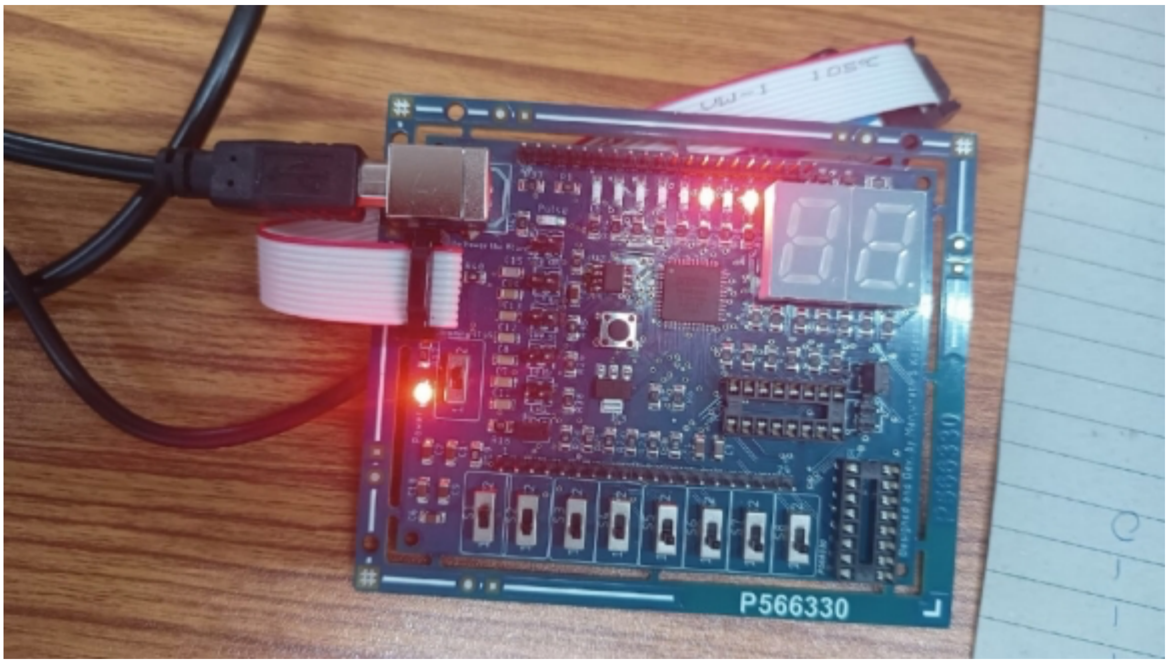


Reset	*Clock*	$Q_{(t-1)}$	$Q_{(t)}$
1	X	X	000
0	↓	000	001
0	↓	001	010
0	↓	010	011
0	↓	011	100
0	↓	100	101
0	↓	101	110
0	↓	110	111
0	↓	111	000

Code:

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

entity q1 is
Port ( CLOCK : in STD_LOGIC;
RESET : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (2 downto 0));
end q1;

architecture Behavioral of q1 is
  signal q_tmp: std_logic_vector(2 downto 0):= "000";
  begin
  process(CLOCK,RESET)
  begin
  if RESET = '1' then
    q_tmp <= "000";
    1
  elsif
falling_edge(CLOCK)
then
    if(q_tmp(0)='0')
then
      q_tmp(0)<='1';
    elsif q_tmp(1)='0'
then q_tmp(1)<= '1';
      q_tmp(0)<='0';
    elsif q_tmp(2)= '0'
then q_tmp(2)<= '1';
      q_tmp(1)<= '0';
      q_tmp(0)<='0';
    else
      q_tmp(2)<= '0';
      q_tmp(1)<= '0';
      q_tmp(0)<='0';
    end if;
  end if;
  end process;
  Q <= q_tmp;
end Behavioral;
```





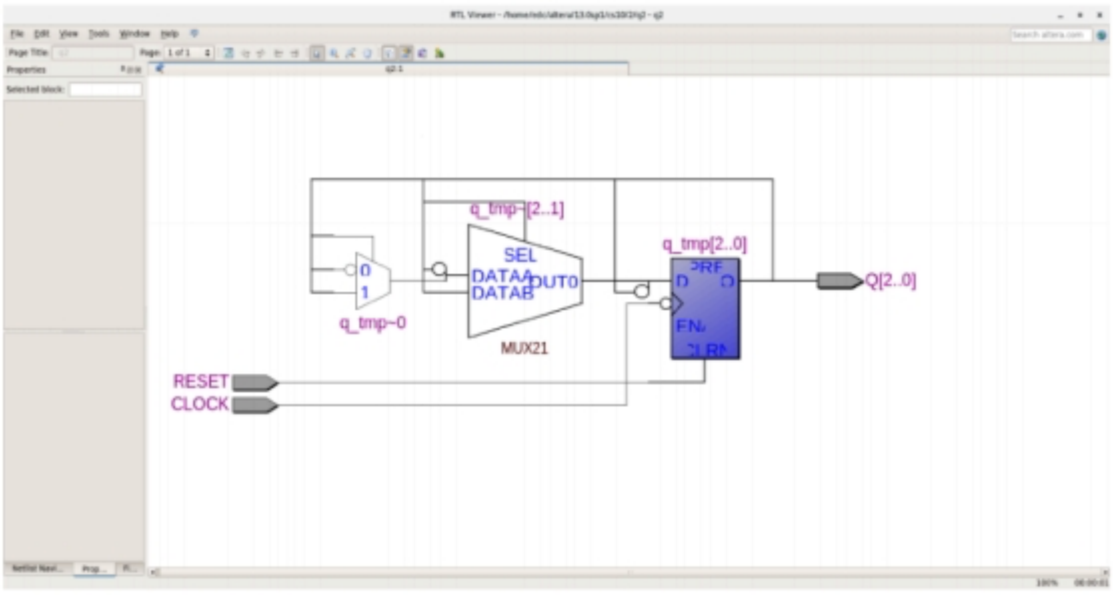
2. Asynchronous Down Counter

$Reset$	$Clock$	$Q_{(t-1)}$	$Q_{(t)}$
1	X	X	111
0	$\downarrow$	111	110
0	$\downarrow$	110	101
0	$\downarrow$	101	100
0	$\downarrow$	100	011
0	$\downarrow$	011	010
0	$\downarrow$	010	001
0	$\downarrow$	001	000
0	$\downarrow$	000	111

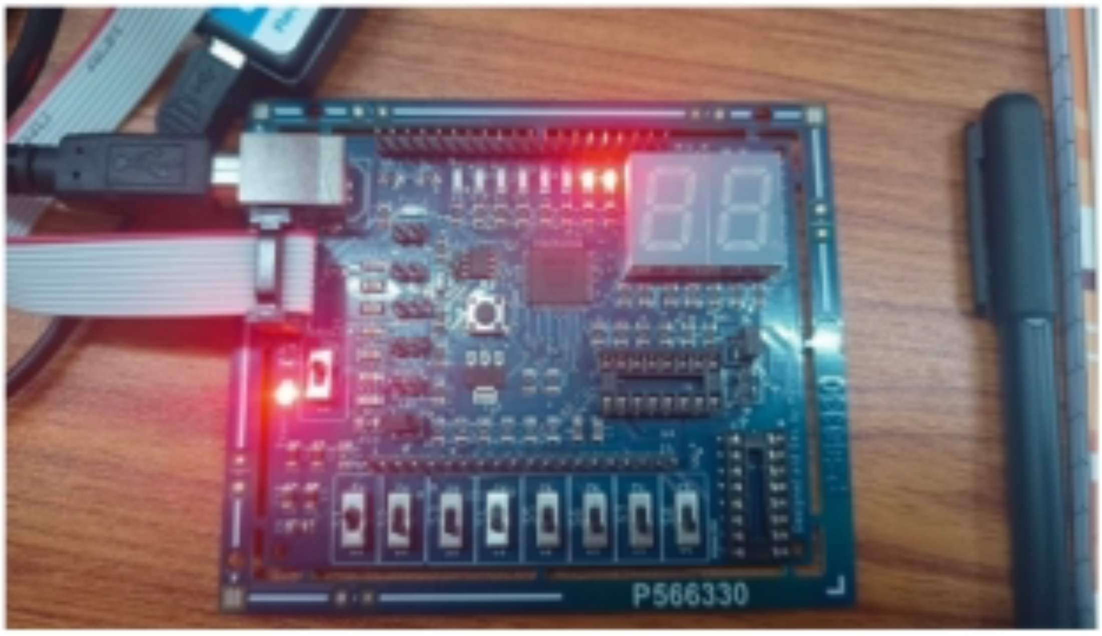
code :

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

entity q2 is
Port ( CLOCK : in STD_LOGIC;
RESET : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (2 downto 0));
end q2;

architecture Behavioral of q2 is
  signal q_tmp: std_logic_vector(2 downto 0):= "111";
  begin
  process(CLOCK,RESET)
    begin
    if RESET = '1' then
      q_tmp <= "000";
    elsif falling_edge(CLOCK) then
      if(q_tmp(0)='1') then
        q_tmp(0)<='0';
      elsif q_tmp(1)='1' then
        q_tmp(1)<= '0';
        q_tmp(0)<='1';
      elsif q_tmp(2)= '1' then
        q_tmp(2)<= '0';
        q_tmp(1)<= '1';
        3
        q_tmp(0)<='1';
      else
        q_tmp(2)<= '1';
        q_tmp(1)<= '1';
```

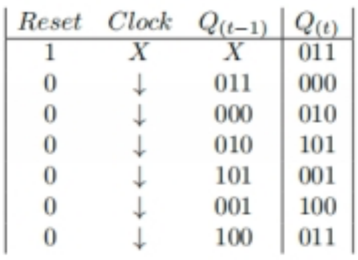
```
            q_tmp(0)<='1';
          end if;
        end if;
      end process;
      Q <= q_tmp;
    end Behavioral;
```



Circuit Snapshots:



3. Sequence Counter

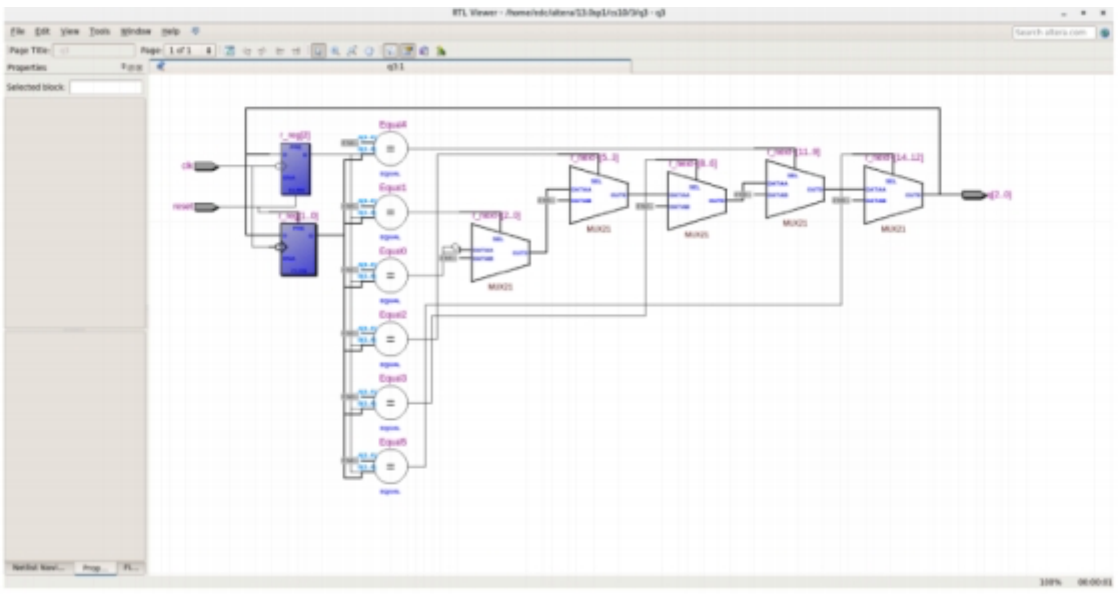


Reset	*Clock*	$Q_{(t-1)}$	$Q_{(t)}$
1	*X*	*X*	011
0	↓	011	000
0	↓	000	010
0	↓	010	101
0	↓	101	001
0	↓	001	100
0	↓	100	011

code :

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


entity q3 is
port (clk, reset: in std_logic;
q: out std_logic_vector (2 downto 0));
end q3;


architecture two_seg_arch of q3 is
  signal r_reg: std_logic_vector (2 downto 0);
  signal r_next: std_logic_vector (2 downto 0);
  Begin
  -- register
  process (clk, reset)
    begin
    if (reset= '1') then
      r_reg <= "011";
      elsif falling_edge(clk) then
      r_reg <= r_next;
    end if;
  end process;
  -- next-state logic
```

```
   r_next <= "011" when
r_reg="100" else
   "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"; -- r_reg="111"
   -- output logic
   q <= r_next;
end two_seg_arch;
```



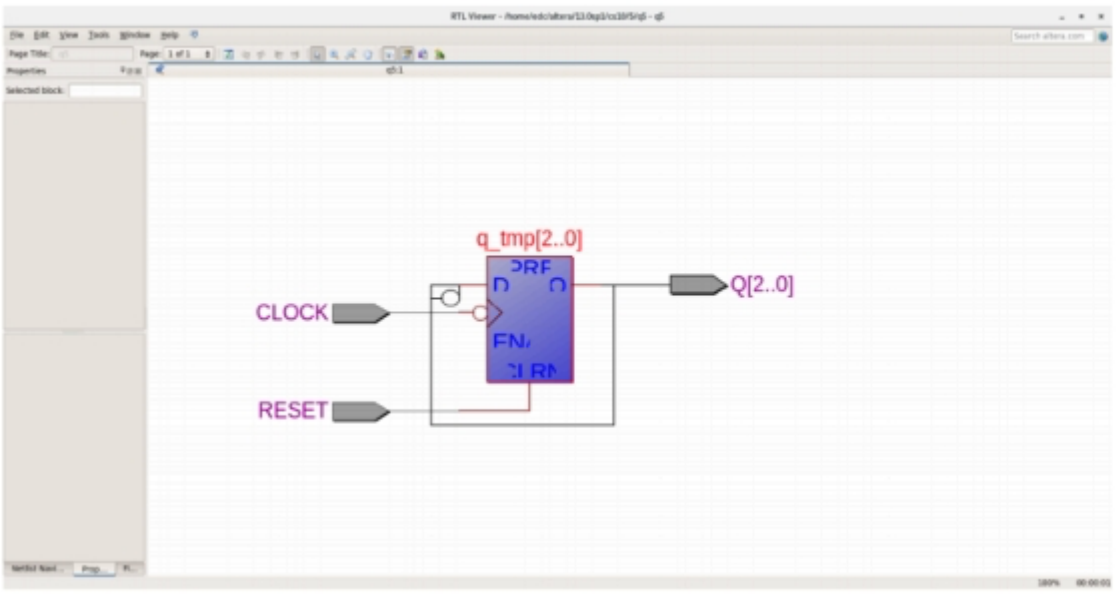
4. 3-bit Ring Counter

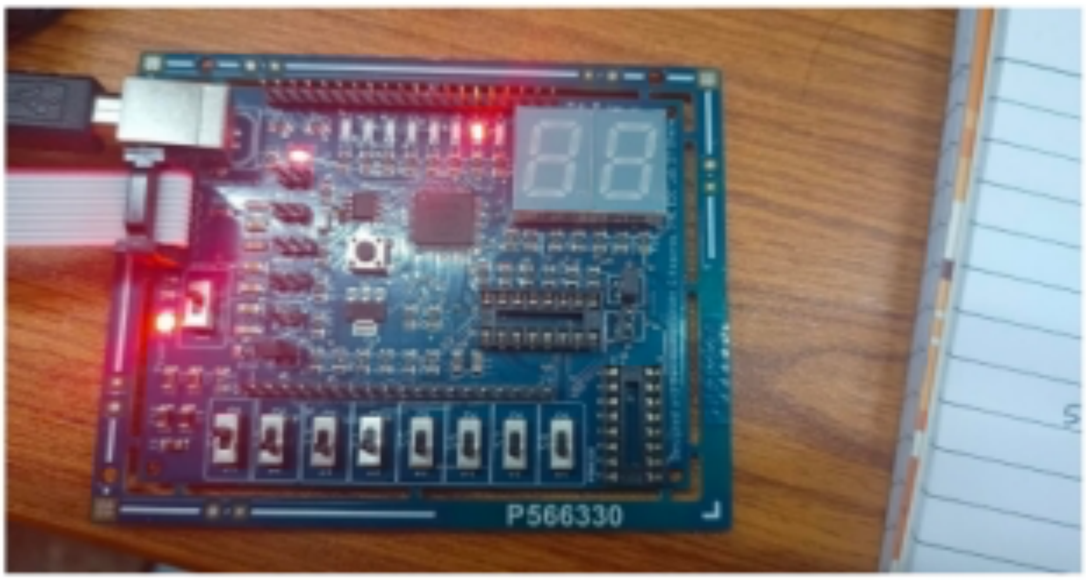
Reset	*Clock*	$Q_{(t-1)}$	$Q_{(t)}$
1	X	X	100
0	↓	100	010
0	↓	010	001
0	↓	001	100

code :

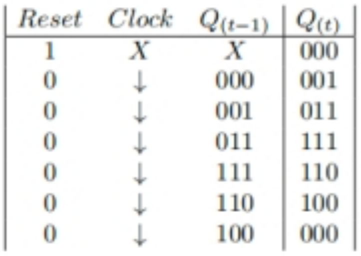
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
6
entity q4 is
Port ( CLOCK : in STD_LOGIC;
RESET : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (2 downto 0));
end q4;
architecture Behavioral of q4 is
signal q_tmp: std_logic_vector(2 downto 0):= "000";
begin
process(CLOCK,RESET)
begin
if RESET = '1' then
q_tmp <= "001";
```

```
elsif falling_edge(CLOCK)
then
q_tmp(1) <= q_tmp(0);
q_tmp(2) <= q_tmp(1);
q_tmp(0) <= q_tmp(2);
end if;
end process;
Q <= q_tmp;
end Behavioral;
```





5. 3-bit Johnson Counter

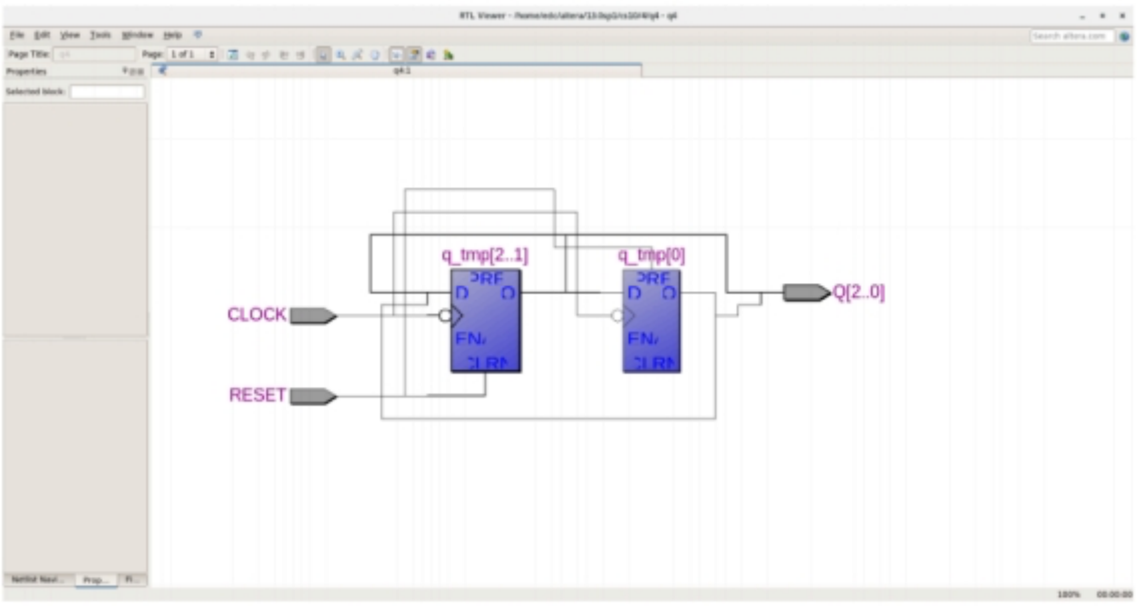


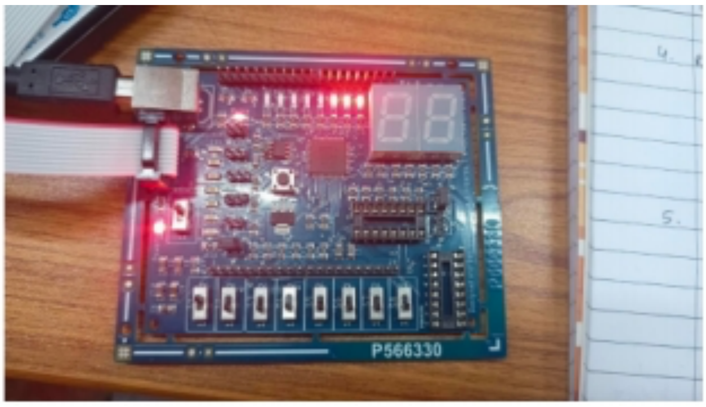
$Reset$	$Clock$	$Q_{(t-1)}$	$Q_{(t)}$
1	X	X	000
0	$\downarrow$	000	001
0	$\downarrow$	001	011
0	$\downarrow$	011	111
0	$\downarrow$	111	110
0	$\downarrow$	110	100
0	$\downarrow$	100	000

code :

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


entity q5 is
Port ( CLOCK : in STD_LOGIC;
RESET : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (2 downto 0));
end q5;


architecture Behavioral of q5 is
  signal q_tmp: std_logic_vector(2 downto 0):= "000";
  begin
  process(CLOCK,RESET)
    begin
    if RESET = '1' then
      q_tmp <= "000";
    elsif falling_edge(CLOCK) then
      q_tmp(1) <= q_tmp(0);
      q_tmp(2) <= q_tmp(1);
      q_tmp(0) <= not q_tmp(2);
    end if;
```

```
        end process;
        Q <= q_tmp;
    end Behavioral;
```





Conclusion:

Writing VHDL codes and dumping those codes in CPLD boards for simulation of circuits for easier implementation of a combination of logic gates.

Thank You.