EXPERIMENT-10

Objective: To design and different types of counters using VHDL.

Resources Required:

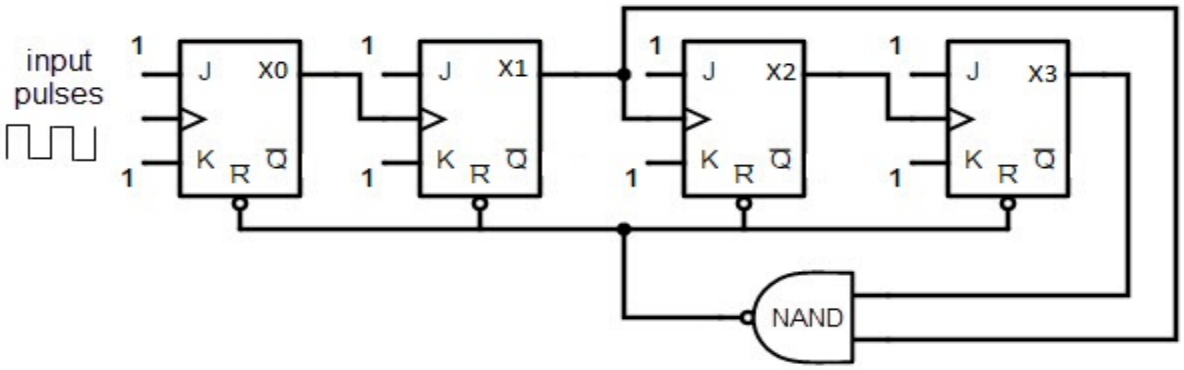
Hardware Requirement: Computer

Software Requirement: XILINX 8.2 Software

Theory:

A) Decade Counter: A binary coded decimal (BCD) is a serial digital counter that counts ten digits .And it resets for every new clock input. As it can go through 10 unique combinations of output, it is also called as "Decade counter". A BCD counter can count 0000, 0001, 0010, 1000, 1001, 1010, 1011, 1110, 1111, 0000, and 0001 and so on.

A 4 bit binary counter will act as decade counter by skipping any six outputs out of the 16 (24) outputs. There are some available ICs for decade counters which we can readily use in our circuit, like 74LS90. It is an asynchronous decade counter



The above figure shows a decade counter constructed with JK flip flop. The J output and K outputs are connected to logic 1. The clock input of every flip flop is connected to the output of next flip flop, except the last one. The output of the NAND gate is connected in parallel to the clear input 'CLR' to all the flip flops. This ripple counter can count up to 16 i.e. 24.

When the Decade counter is at REST, the count is equal to 0000. This is first stage of the counter cycle. When we connect a clock signal input to the counter circuit, then the circuit will count the binary sequence. The first clock pulse can make the circuit to count up to 9 (1001). The next clock pulse advances to count 10 (1010).

Then the ports X1 and X3 will be high. As we know that for high inputs, the NAND gate output will be low. The NAND gate output is connected to clear input, so it resets all the flip flop stages in decade counter. This means the pulse after count 9 will again start the count from count 0.

Truth Table:

Input Pulses	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
0	0	0	0	0 (resets)

VHDL Code:

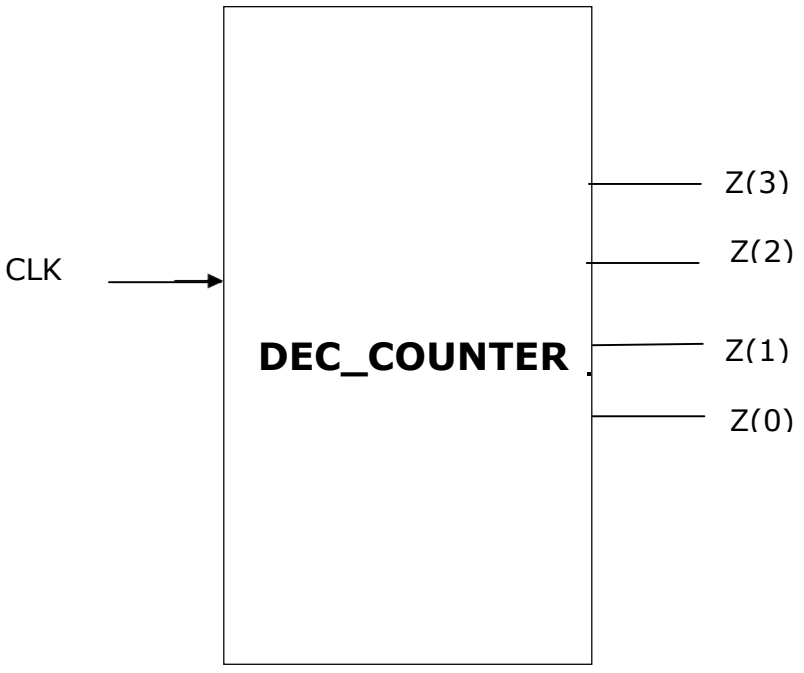
```
COMPONENT JK:-
library ieee;
use ieee.std_logic_1164.all; entity JK is
port(J,K,clk:    in    std_logic;Q:inout    std_logic:='0';Qb:inout
std_logic:='1'); end JK;
architecture ff of JK is begin
process(J,K,clk) variable t,tb: std_logic; begin
t:=Q;
tb:=Qb;
if (clk='0'and clk'event) then if(J='0'and K='0') then t:=t;tb:=tb;
elsif(J='0'and K='1') then t:='0';tb:='1';
elsif(J='1'and K='0') then t:='1';tb:='0'; elsif(J='1'and K='1') then
t:=not t;tb:=not tb; end if;
end if; Q<=t;
Qb<=tb; end process;

end ff;
```

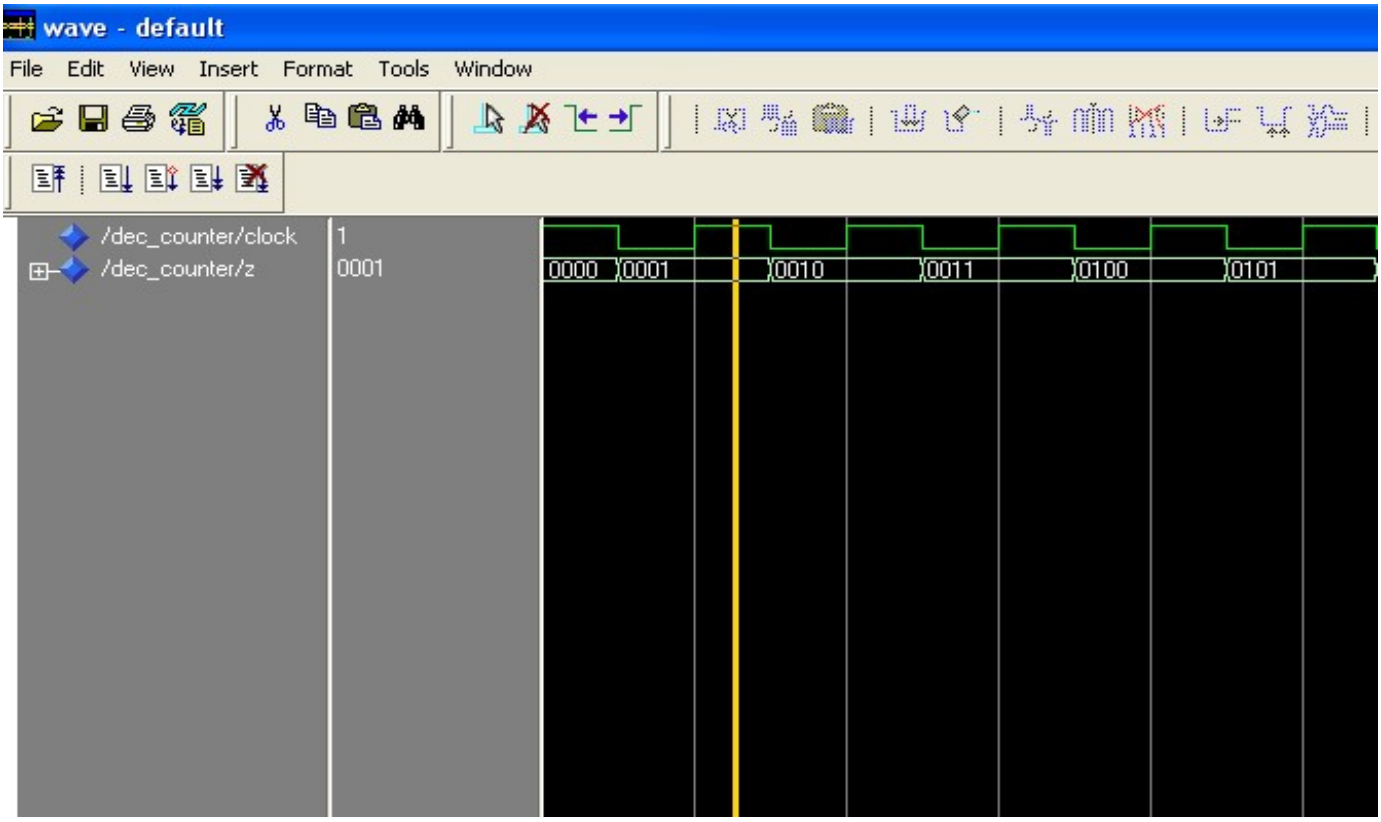
TOP MODULE:-

```
library ieee;
use ieee.std_logic_1164.all; entity dec_counter is
port(clock:in std_logic;z: inout std_logic_vector(3 downto
0):="0000"); end dec_counter;
architecture counter of dec_counter is component JK
port(J,K,clk: in std_logic;Q:inout std_logic:='0';Qb:inout
std_logic:='1'); end component;
component and2
port(a,b:in std_logic;c:out std_logic); end component;
signal s1,s2:std_logic; signal s:std_logic:='1'; begin
JK1: JK port map(s,s,clock,z(0),open); JK2: JK port
map(s2,s,z(0),z(1),open);
JK3: JK port map(s,s,z(1),z(2),open);
X1: and2 port map(z(2),z(1),s1);
JK4: JK port map(s1,s,z(0),z(3),s2); end counter;
```

Output:

RTL Schematic:





B) 3-Bit Updown Counter: The up/Down counter is also known as the bidirectional counter which is used to count in any direction based on the condition of the input control pin. These are used in different applications to count up from zero to provide a change within the output condition on attaining a fixed value & others count down from a fixed value to zero to give an output condition change. There are some types of counters like TTL 74LS190 & 75LS191 which can function in both up & down count mode based on the condition of an input pin of up/down count mode.

Truth Table:

INPUT	PRESENT STATE			NEXT STATE		
UP/ DOWN	q2	q1	q0	Q2	Q1	Q0
0	0	0	0	1	1	1
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	1	0

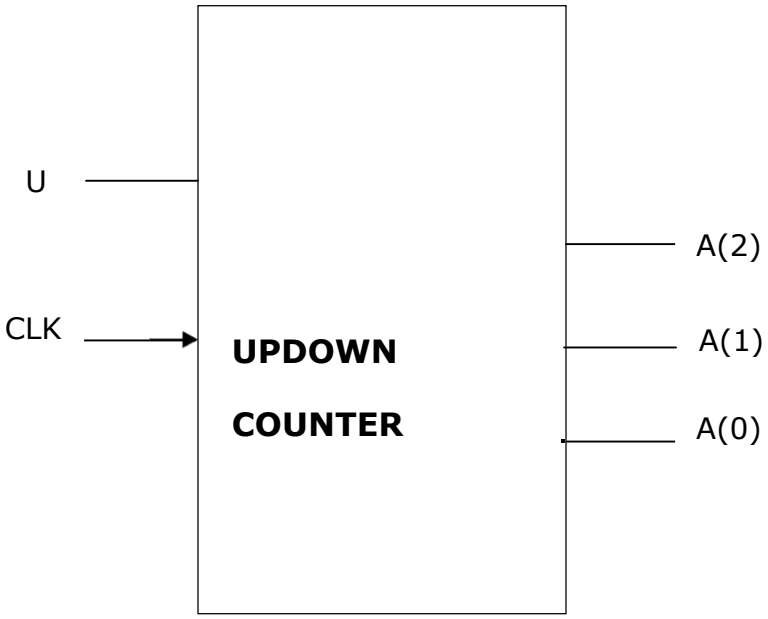
0	1	0	0	0	1	1
0	1	0	1	1	0	0
0	1	1	0	1	0	1
0	1	1	1	1	1	0
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	0	0	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	0	0	0

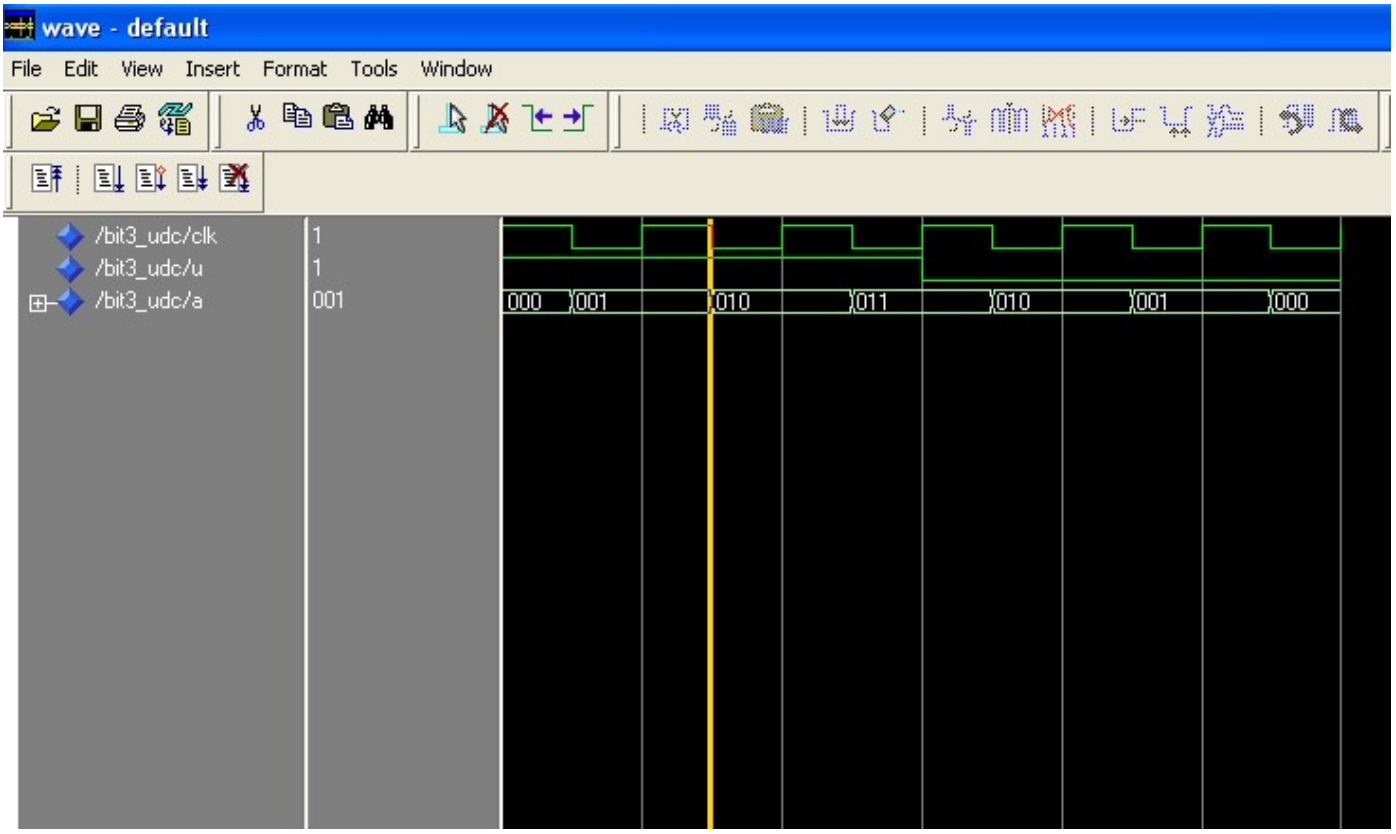
VHDL Code:

```
library ieee;
use ieee.std_logic_1164.all; use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all; entity bit3_udc is
port(clk,u:in    std_logic;a:    inout    std_logic_vector(2    downto
0):="000"); end bit3_udc;
architecture beh of bit3_udc is begin
process(clk,a,u)
variable t: std_logic_vector(2 downto 0); begin

t:=a;
if clk='0' and clk'event then if u='1' then t:= t+"001"; elsif u='0'
then t:= t+"111"; end if;
end if; a<=t;
end process; end beh;
```

Output:

RTL Schematic:





Results:VHDL codes of different counters are simulated & synthesized.