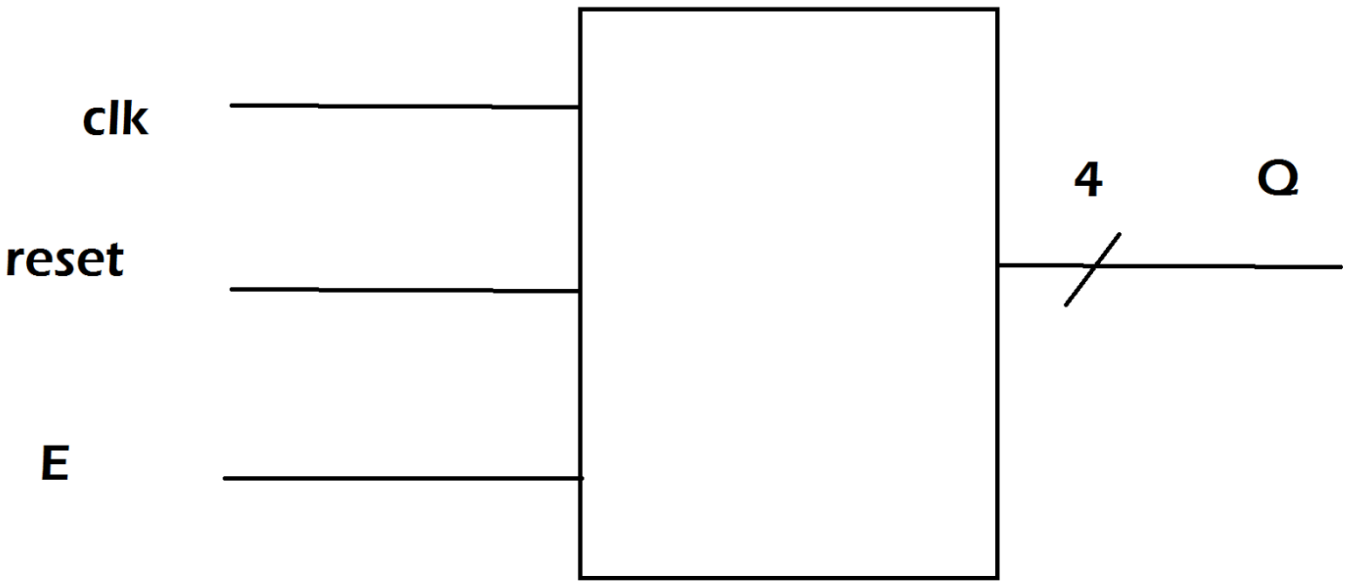
VHDL LANGURAGE

FLIP-FLOPS, REGISTERS, COUNTERS, ANDASIMPLEPROCESSOR

DR.TIN THET NWE1

4-BIT UP COUNTER

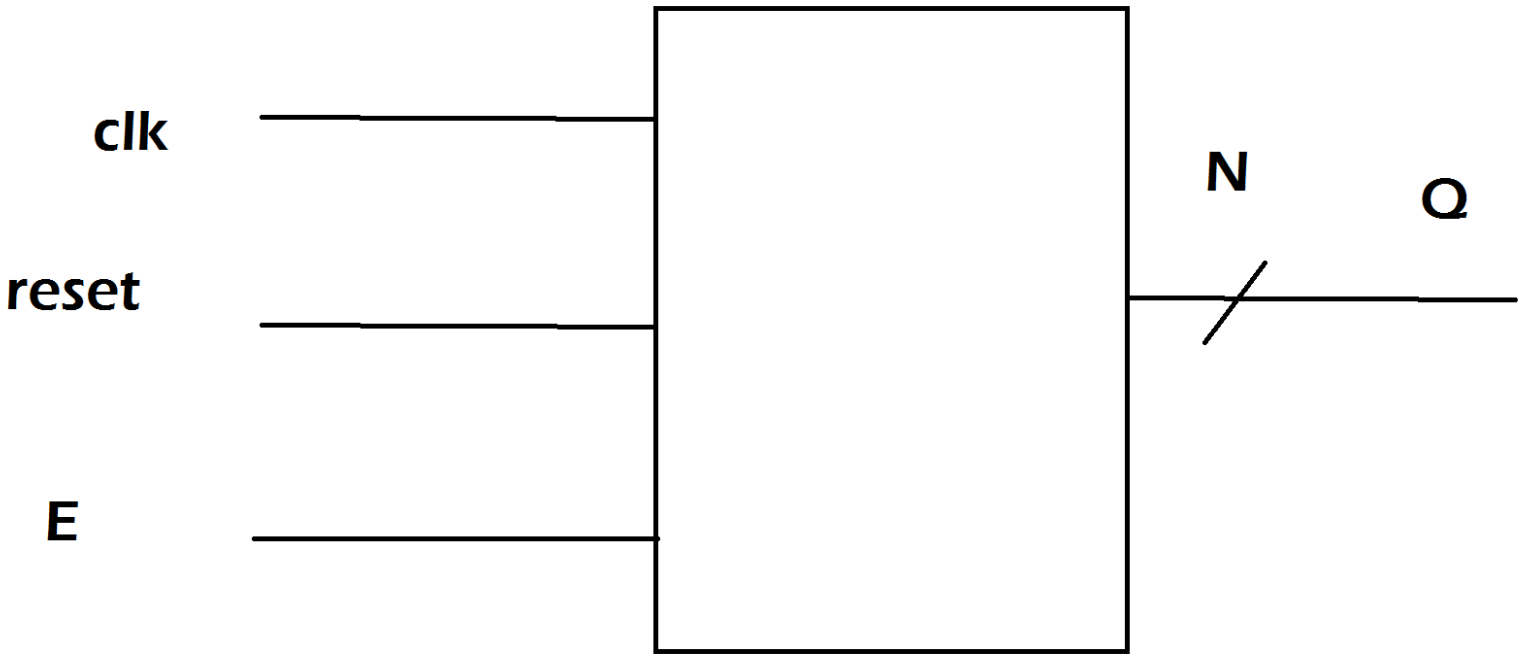


UP COUNTER

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY upcount IS
    PORT ( Clock, Resetn, E   : IN     STD_LOGIC ;
           Q                  : OUT  STD_LOGIC_VECTOR (3 DOWNTO 0)) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
    SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN
    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            Count <= "0000" ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            IF E = '1' THEN
                Count <= Count + 1 ;
            ELSE
                Count <= Count ;
            END IF ;
        END IF ;
    END PROCESS ;
    Q <= Count ;
END Behavior ;
```

Figure 7.52 Code for a four-bit up-counter.

N-BIT UP COUNTER



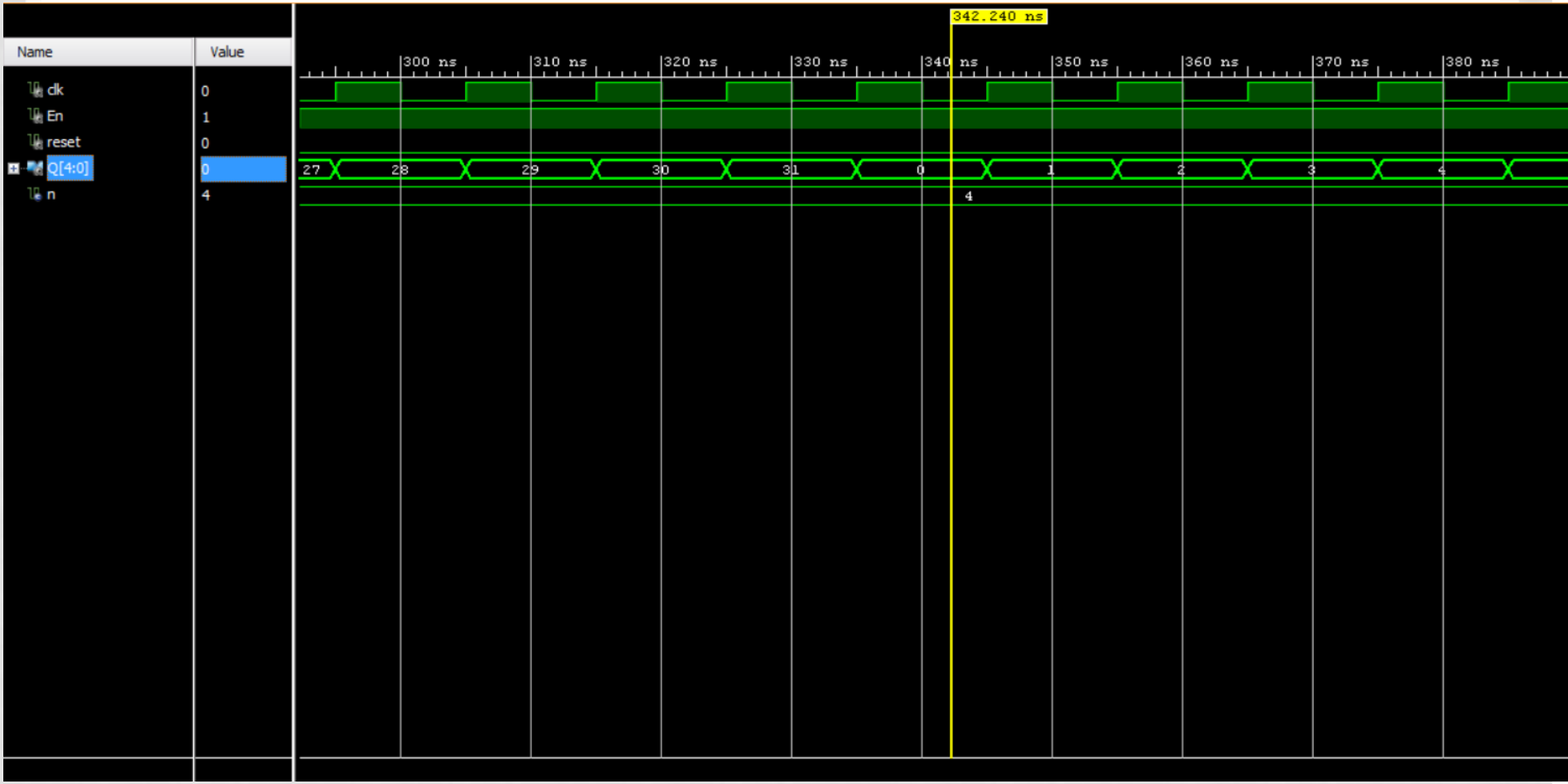
N-BIT UP COUNTER

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


entity n_bit_upcounter is
    generic ( n:integer:=3);
    Port ( clk : in STD_LOGIC;
               En : in STD_LOGIC;
           reset : in STD_LOGIC;
  Q : out STD_LOGIC_VECTOR (n downto 0));
end n_bit_upcounter;

architecture Behavioral of n_bit_upcounter  is
signal count :  std_logic_vector ( n downto 0);
begin
process ( reset,clk,En )
begin
if reset ='1' then
    count<= (others=>'0');
 elsif  rising_edge (clk)  then
    if En ='1' then
        count<= count+1;
    else
        count <= count;
    end if;
 end if;
 end process;

 Q<= count;

 end Behavioral;
```

N=3


0.500 ns
Name
Value
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
clk
0
En
0
reset
1
Q[3:0]
0
n
3
0 1 2 3 4 5 6 7 8 9 a b c d e f 0 1 2 3 4 5 6 7 8 9 a b c d e f
3

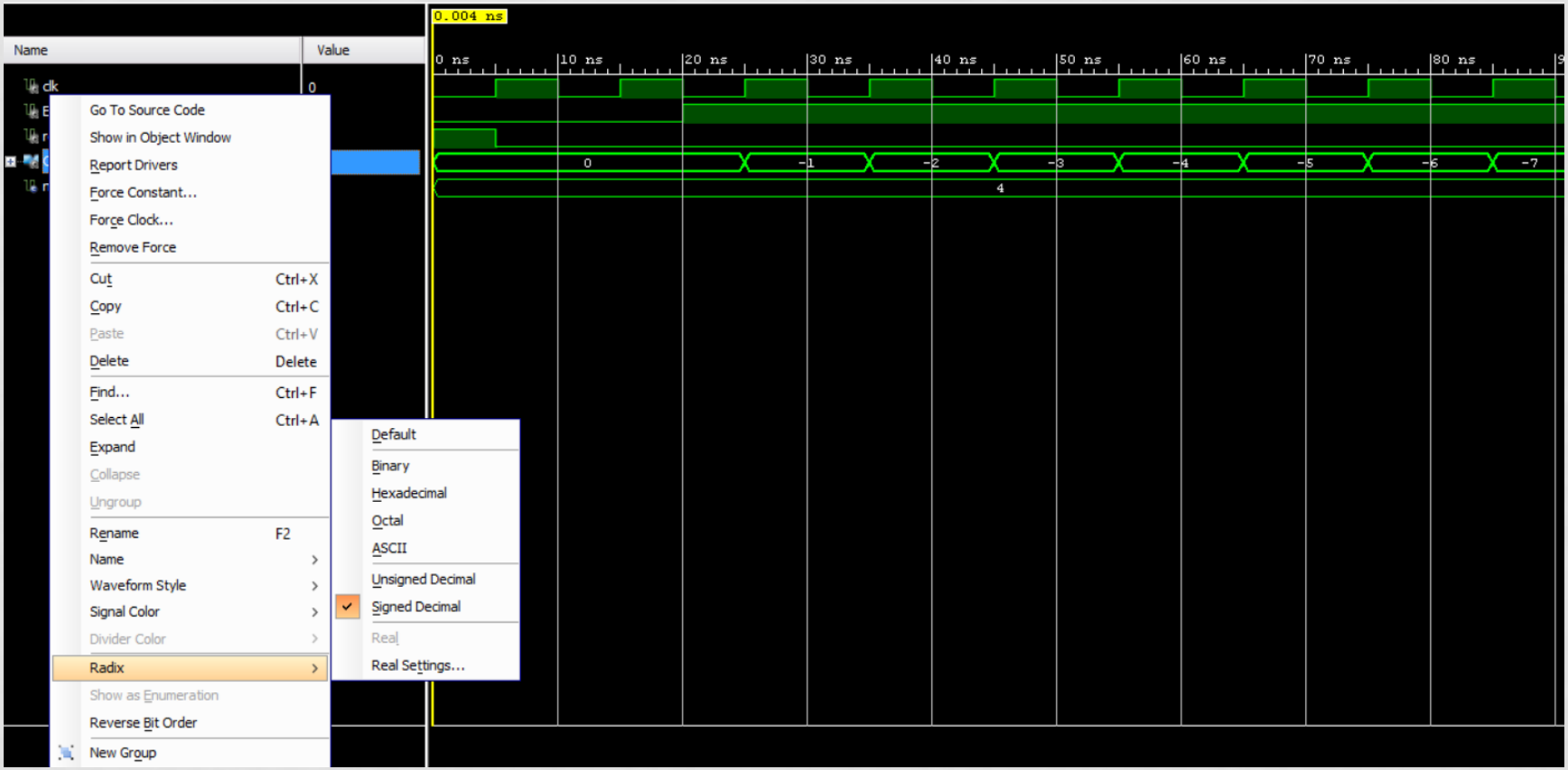
N=4


342.240 ns
Name
Value
300 ns
310 ns
320 ns
330 ns
340 ns
350 ns
360 ns
370 ns
380 ns
clk
0
En
1
reset
0
Q[4:0]
0
n
4
27
28
29
30
31
0
1
2
3
4
4

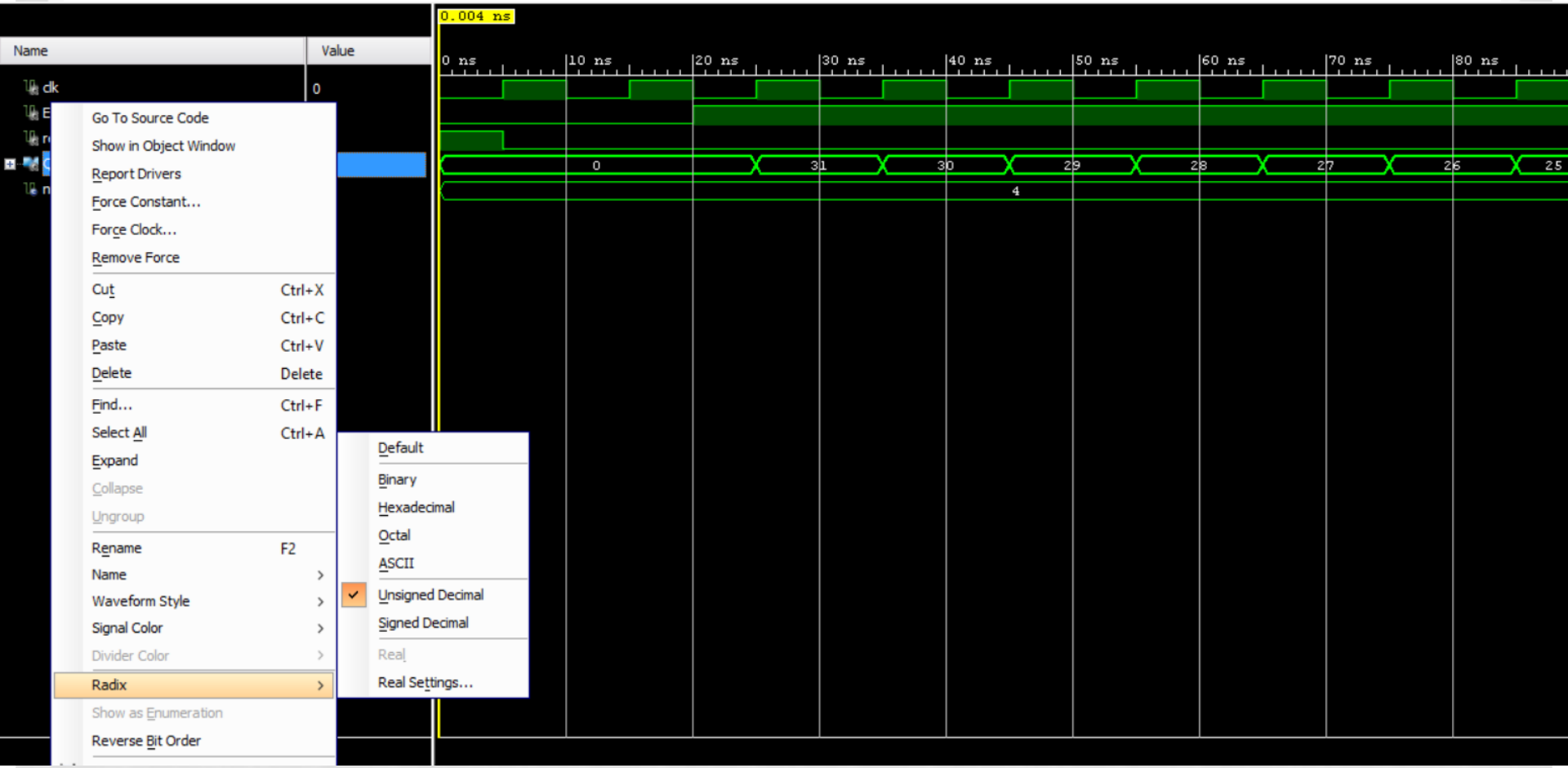
N-BIT DOWN COUNTER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
entity n_bit_dwn_counter is
    generic ( n:integer:=4);
    Port ( clk : in STD_LOGIC;
                En : in STD_LOGIC;
           reset : in STD_LOGIC;
  Q : out STD_LOGIC_VECTOR (n downto
0));
end n_bit_dwn_counter;

architecture Behavioral of
n_bit_dwn_counter  is
signal count :  std_logic_vector ( n downto
0);
begin
process ( reset,clk,En )
begin
if reset ='1' then
    count<= (others=>'0');
elsif  rising_edge (clk)  then
            if En ='1' then
                        count<= count-1;
            else
                        count <= count;
            end if;
 end if;
 end process;
 Q<= count;

 end Behavioral;
```

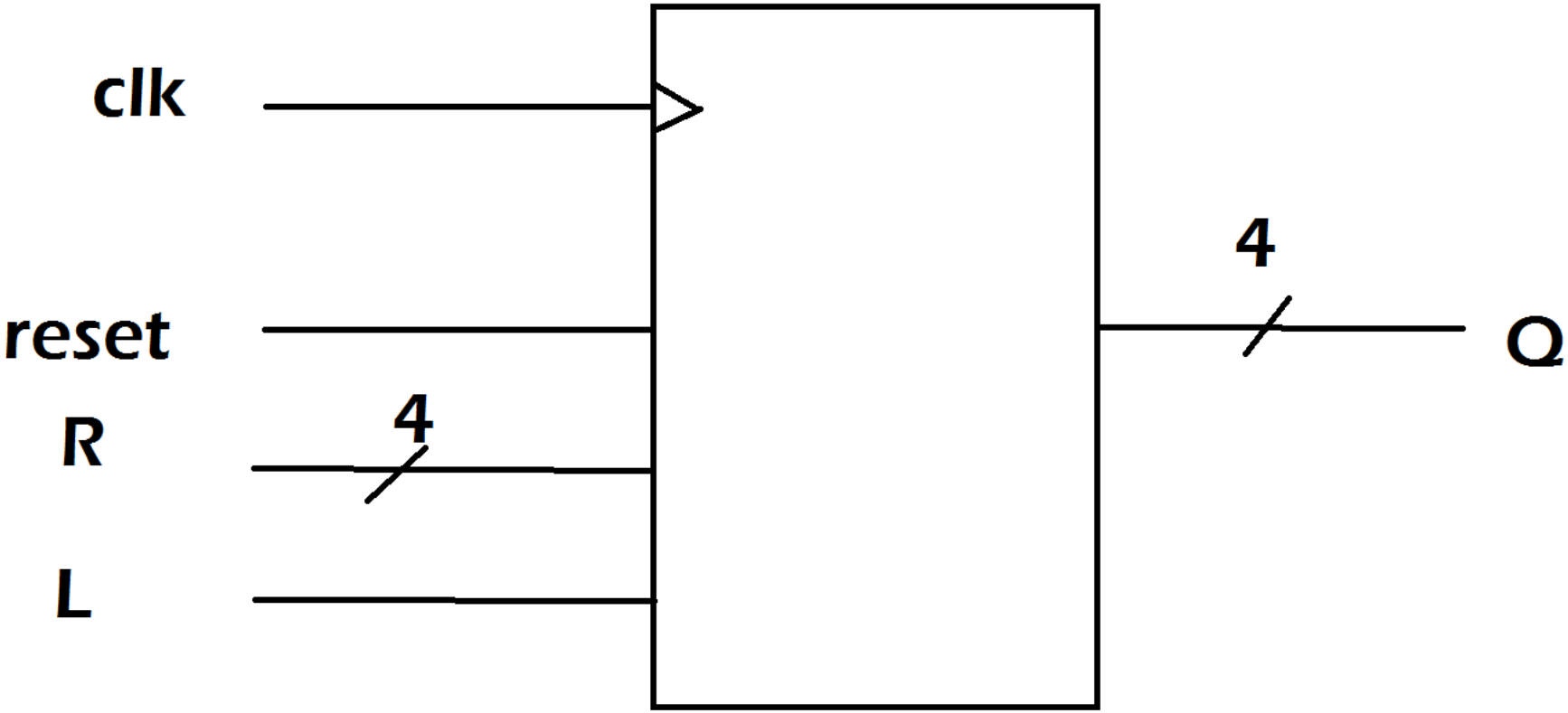
N-BIT DOWN COUNTER



N-BIT DOWN COUNTER



PARALLEL LOAD UP COUNTER



PARALLEL LOAD UP COUNTER

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity up_counter is
generic ( n: integer :=3);
port (
  R : in std_logic_vector ( n downto 0) ;
  clk,reset,L: in std_logic;
  Q: out std_logic_vector ( n downto 0)    );




end up_counter;

architecture arch of up_counter is
signal count : std_logic_vector( n downto
0);
begin
process ( clk,reset)
begin
if  reset ='1' then
    count <=( others =>'0');
 elsif rising_edge ( clk) then
    if L='1' then
        count<= R;
    else
        count <= count+1;
    end if;

end if;
end process;
Q<=count;

end arch;
```

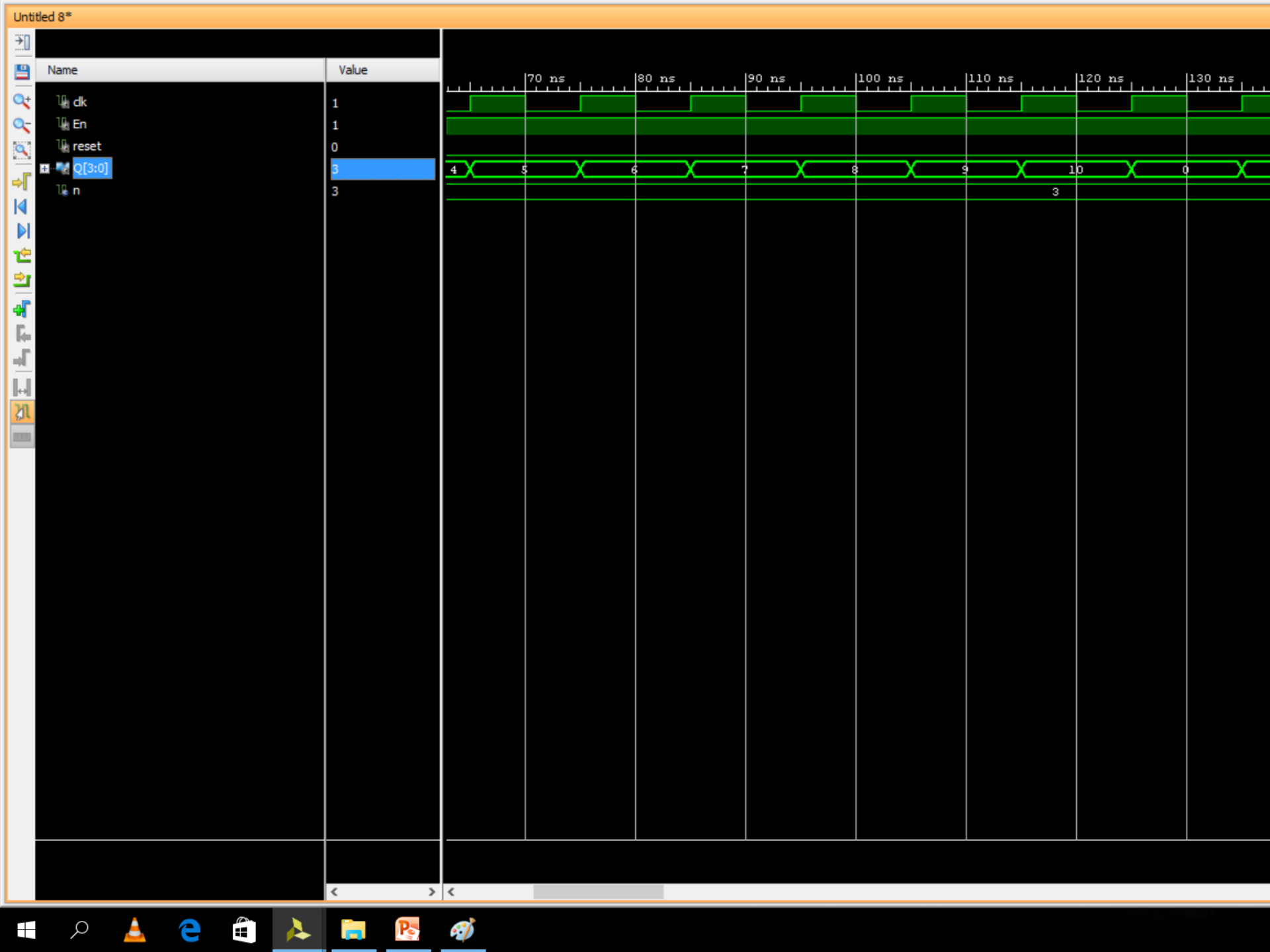
PARALLEL LOAD UP COUNTER (TEST BENCH)

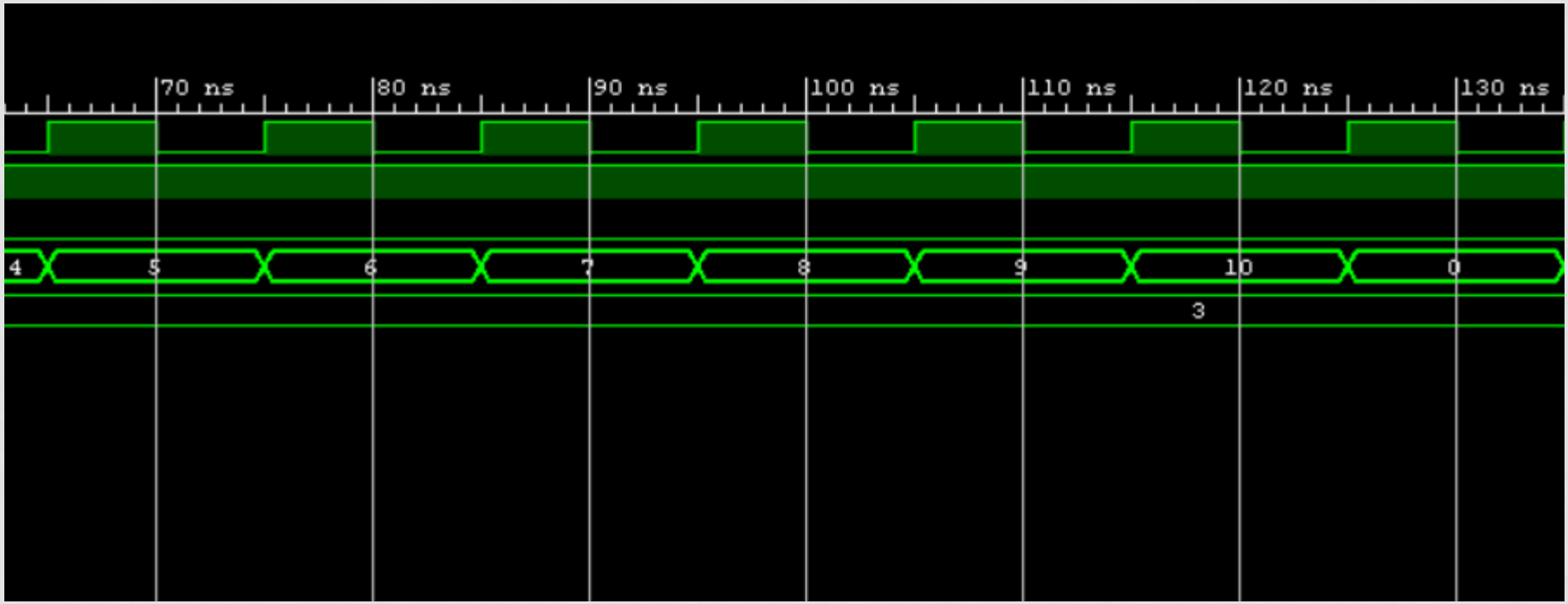
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity sim_counter is
generic ( n: integer:=3);
end sim_counter;

architecture Behavioral of sim_counter is
component up_counter is
generic ( n: integer :=3);
port (
  R : in std_logic_vector ( n downto 0) ;
  clk,reset,L: in std_logic;
  Q: out std_logic_vector ( n downto 0)    );



end component;
signal  R :  std_logic_vector ( n downto 0) ;
  signal clk,reset,L:  std_logic;
 signal  Q:  std_logic_vector ( n downto 0) ;
begin

U: up_counter
generic map ( n=>n)
port map ( R=>R, clk=>clk, reset => reset ,L=>L ,  Q =>Q);

process
begin
clk<='0' ; wait for 5 ns;
clk<='1'; wait for 5 ns;
end process;

process
begin
reset <='1' ; wait for 5 ns;
reset <='0'; wait ;

end process;

process
begin
L <='0' ; wait for 20 ns;
L <='1'; wait for 15 ns ;
L <= '0' ; wait ;
end process;
process
begin
R <= "1010"; wait;
end process;
end Behavioral;
```


0.004 ns
Value
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
0
1
0
10
0
3
10
0
1
2
10
11
12
13
3
im_counter/n

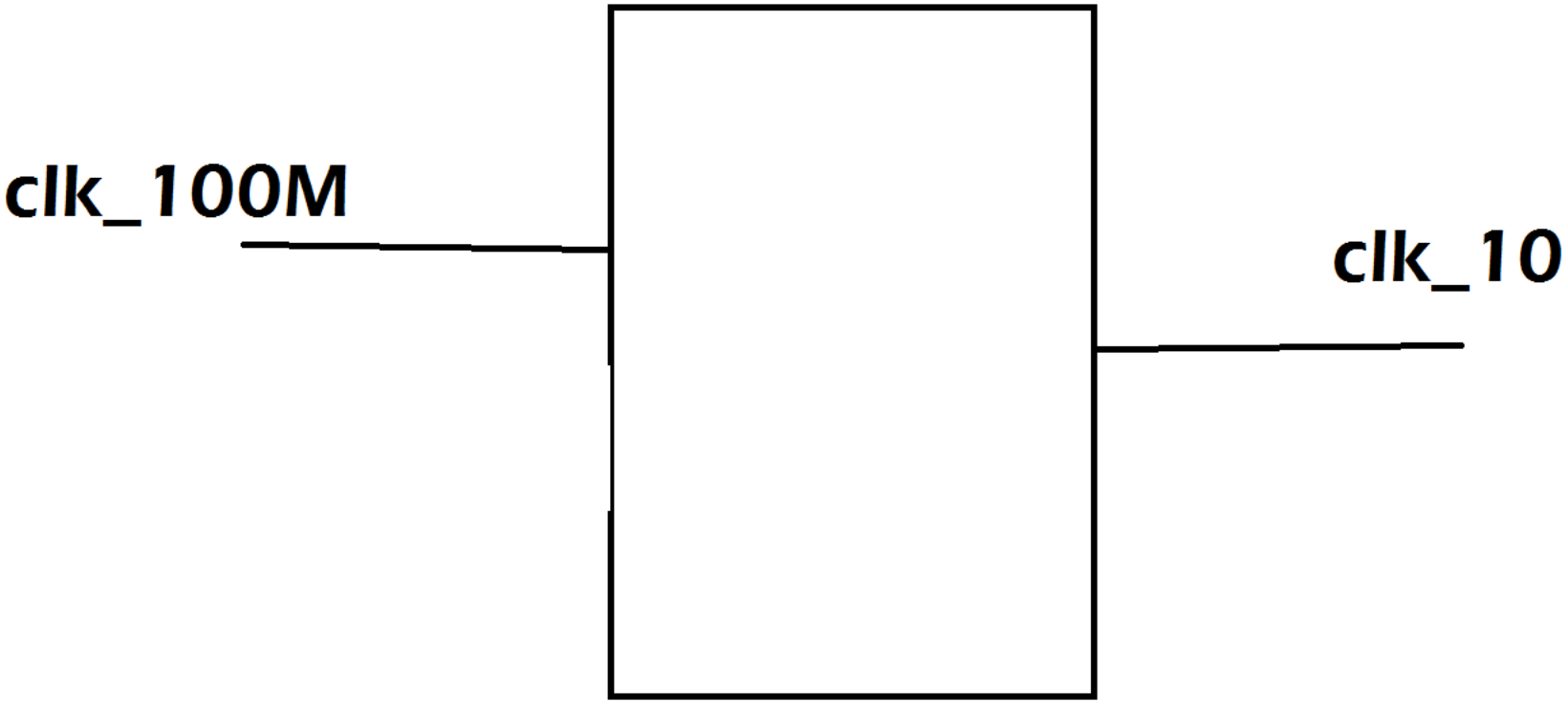
MODULUS-10 UP COUNTER

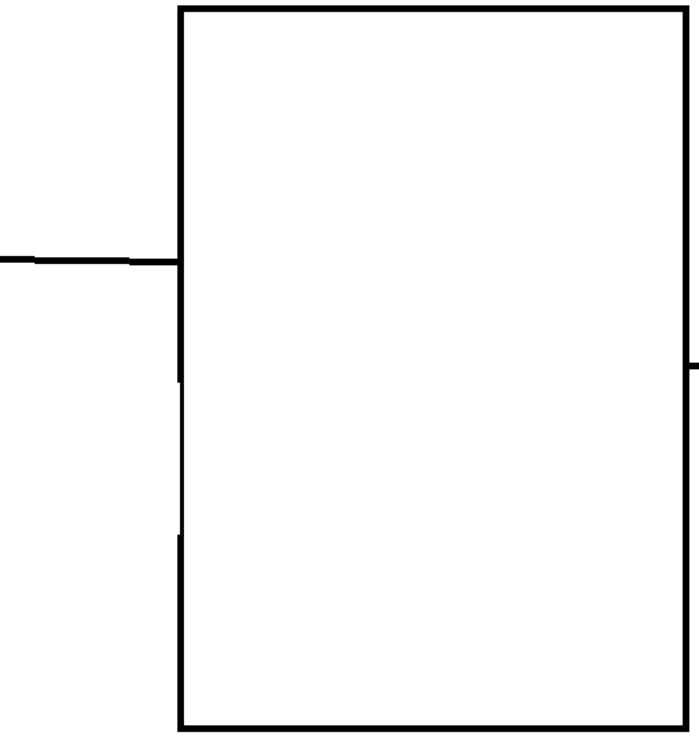
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;


entity n_bit_upcounter is
    generic ( n:integer:=3);
    Port ( clk : in STD_LOGIC;
               En : in STD_LOGIC;
           reset : in STD_LOGIC;
  Q : out STD_LOGIC_VECTOR (n downto
0));
end n_bit_upcounter;

architecture Behavioral of
n_bit_upcounter  is
signal count :  std_logic_vector ( n
downto 0);
begin


process ( reset,clk,En )
begin
if reset ='1' then
    count<= (others=>'0');
 elsif  rising_edge (clk)  then
    if En ='1' then
              if count="1010" then
                 count<= (others=>'0');
              else
                 count <= count+1;
              end if;
     end if;
end if;
 end process;

 Q<= count;

 end Behavioral;
```


Untitled 8*
Name
Value
clk
En
reset
Q[3:0]
n
1
1
0
3
3
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
4
5
6
7
8
9
10
0
3


70 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
4
5
6
7
8
9
10
0
3

CLOCK DIVIDER





clk_10

T = 1/10 = 100ms

clk 100M

T = 1/100M

=10M

5ns

5ns

00M/(10*2) -1

,999,999

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

entity clk_divider is
    Port ( clk_100M : in STD_LOGIC;
           clk_10 : out STD_LOGIC);
end clk_divider;

architecture Behavioral of clk_divider is
signal cnt:integer range 0 to 4999999:=0;
signal clk1:std_logic:='0';
begin
process ( clk_100M)
begin
if rising_edge ( clk_100M)  then
    if  cnt = 4999999 then
            --( 100M/(10*2)-1)
        cnt <=0;
        clk1<= not clk1;
     else
         cnt <= cnt+1;
    end if;
end if ;
end process;
clk_10 <= clk1;
end Behavioral;
```

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

entity sim is
end sim;

architecture Behavioral of sim is

component clk_divider is
    Port ( clk_100M : in STD_LOGIC;
           clk_10 : out STD_LOGIC);
end component;


signal clk_100M:std_logic;
signal clk_10: std_logic;

begin

UU:clk_divider
port map
( clk_100M => clk_100M ,
 clk_10 => clk_10);

process
begin
clk_100M  <='0'; wait for 5 ns;
clk_100M  <='1'; wait for 5 ns;
end process;

end Behavioral;
```