4)
Design an 8 bit counter in both of the Verilog and VHDL languages and simulate and test in the Synopsys environment.   You should create a behavior and test it with the appropriate test bench.

Ans:

8 bit counter in Verilog:

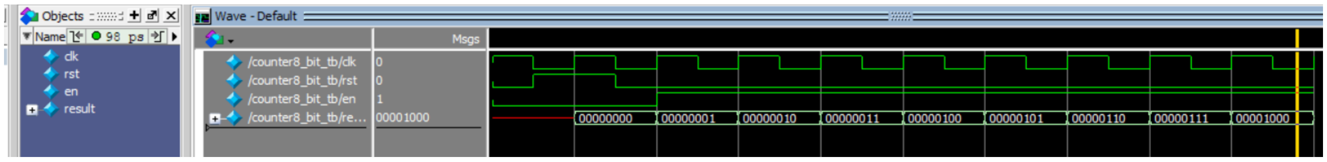
```
module counter(clk, rst, en, result);
input clk ;
input rst ;
input en ;
output reg [7:0] result ;
wire clk ;
wire rst ;
wire en ;
reg [7:0] Result ;
always @ (posedge clk)
begin
if (rst == 1'b1)
begin
Result <= 8'b00000000;
end
else
begin
if (en == 1'b1)
begin
Result <= Result + 1;
end
else
result <= Result;
end
end
endmodule
```

Test bench:

```
module counter8_bit_tb;
reg clk;
reg rst;
reg en;
wire [7:0] result;
counter uut (.clk(clk),.rst(rst),.en(en),.result(result));
initial begin
clk = 0;
```

```
rst = 0;
en = 0;
$display("time\tclk\trst\ten\tresult");
$monitor("%g\t%b\t%b\t%b\t%b",$time,clk,rst,en,result);
clk = 1;
#5 rst = 1;
#10 rst = 0;
#5 en = 1;
#100 en= 0;
end
always begin
#5 clk = ~clk;
end
endmodule
```

Wave form:



8 bit counter in VHDL:

Program:

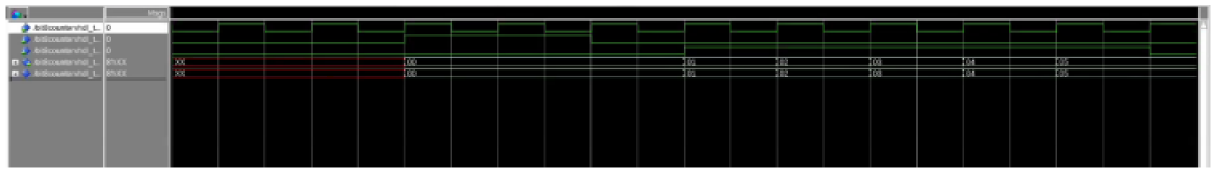
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity bit8countervhdl is
port( clock: in std_logic; reset: in std_logic; enable: in std_logic; result: out std_logic_vector(7
downto 0));
end bit8countervhdl;
architecture Behavioral of bit8countervhdl is
signal count: std_logic_vector(7 downto 0);
begin
process(clock, enable, reset)
begin
if (reset = '1') then
count <= "00000000";
elsif (rising_edge(clock)) then
if (enable = '1') then
count <= count + '1';
end if;
end if;
end process;
```

```
result <= count;
end Behavioral;
```

Test Bench:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity bit8countervhdl_tb IS
end bit8countervhdl_tb;
architecture behavior OF bit8countervhdl_tb IS
component bit8countervhdl
port(
clock : IN std_logic;
reset : IN std_logic;
enable : IN std_logic;
result : OUT std_logic_vector(7 downto 0)
);
end component;
signal clock : std_logic := '0';
signal reset : std_logic := '0';
signal enable : std_logic := '0';
signal result : std_logic_vector(7 downto 0);
begin
uut: bit8countervhdl port map (
clock => clock,
reset => reset,
enable => enable,
result => result
);
clk_p: process
begin
wait for 1 ns; clock<= not clock;
end process;
stim_p: process
begin
wait for 5 ns; reset <= '1';
wait for 4 ns; reset <= '0';
wait for 2 ns; enable <= '1';
wait for 10 ns; enable <= '0';
wait;
end process;
end;
```

Wave form:



3)
Write a generalized testbench function which enumerates all of the possible combinatorial input states for a given number of input pins.   The number of input pins is to be a parameter or generic.

a) Write the function in Verilog
b) Write the function in VHDL

Ans:
a)
```
module pcis_tb;
parameter no_ip= 4;
reg n;
integer i, no_o;
reg [no_ip-1:0] r;
pcis uut (.n(n));
initial begin
n = 0;
r = 0;
$monitor ($time,"r=%b",r,"i=%b",i);
no_o=2**(no_ip);
for(i=0; i<no_o; i=i+1)
begin
if(i == 0)
r = 0;
else
r=r+1;
#10;
end
end
endmodule
```

b)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library STD;
use STD.TEXTIO.ALL;
use IEEE.STD_LOGIC_TEXTIO.ALL;
use IEEE.NUMERIC_STD.ALL;
entity pcis_tb is
end pcis_tb;
architecture behavior of pcis_tb is
constant k : integer := 4;
component pcis is
generic(no_ip : integer:= k);
port( r : in STD_LOGIC_VECTOR (no_ip-1 downto 0));
end component;
signal r : STD_LOGIC_VECTOR (k-1 downto 0) := (others =>'0');
constant no_o: integer:= 2**k;
begin
uut: pcis generic map (no_ip => k) port map (r => r);
process
begin
wait for 100 ns;
for i in 0 to no_o-1 loop
r <= STD_LOGIC_VECTOR(unsigned(r) + 1);
wait for 10 ns;
end loop;
wait;
end process;
end;
```