۱. واحد رجیستر ۸ بیتی

کد برنامه :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity Reg_8bits is
    Port ( Din : in  STD_LOGIC_VECTOR (7 downto 0);
           Dout : out  STD_LOGIC_VECTOR (7 downto 0);
           Load : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           Reset : in  STD_LOGIC);
end Reg_8bits;

architecture Behavioral of Reg_8bits is

signal Reg1 : std_logic_vector (7 downto 0) ;
begin

process(clk)
begin
if (clk'event and clk = '1') then
   if Reset = '0' then
      Reg1 <= "00000000";
   elsif load = '1' then
      reg1 <= Din ;
   end if ;
   Dout <= Reg1 ;

end if;
end process;

end Behavioral;


```

ریست به صورت اکیتو لو تعریف شده است و اولویت با آن می باشد. اگر پایه لود فعال شده باشد ورودی را در رجیستر خواهد ریخت. چون همیشه خروجی رجیستر مقدار آن را نمایش می دهد پس از سیکل ریختن لود مقدار خروجی نیز تغییر می کند.

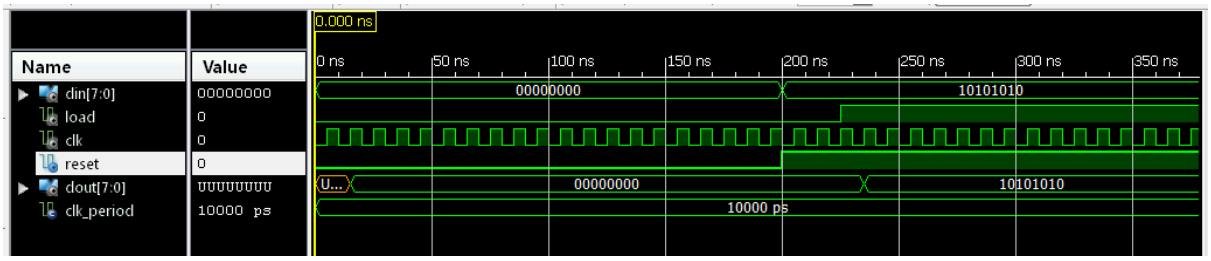
کد شبیه سازی :

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4
5  ENTITY TEST IS
6  END TEST;
7
8  ARCHITECTURE behavior OF TEST IS
9
10     -- Component Declaration for the Unit Under Test (UUT)
11
12     COMPONENT Reg_8bits
13     PORT(
14          Din : IN  std_logic_vector(7 downto 0);
15          Dout : OUT  std_logic_vector(7 downto 0);
16          Load : IN  std_logic;
17          clk : IN  std_logic;
18          Reset : IN  std_logic
19         );
20     END COMPONENT;
21
22
23    --Inputs
24    signal Din : std_logic_vector(7 downto 0) := (others => '0');
25    signal Load : std_logic := '0';
26    signal clk : std_logic := '0';
27    signal Reset : std_logic := '0';
28
29    --Outputs
30    signal Dout : std_logic_vector(7 downto 0);
31
32    -- Clock period definitions
33    constant clk_period : time := 10 ns;
```

```
34
35  BEGIN
36
37     -- Instantiate the Unit Under Test (UUT)
38     uut: Reg_8bits PORT MAP (
39            Din => Din,
40            Dout => Dout,
41            Load => Load,
42            clk => clk,
43            Reset => Reset
44          );
45
46     -- Clock process definitions
47     clk_process :process
48     begin
49        clk <= '0';
50        wait for clk_period/2;
51        clk <= '1';
52        wait for clk_period/2;
53     end process;
54
55
56     -- Stimulus process
57     stim_proc: process
58     begin
59        -- hold reset state for 100 ns.
60        wait for 100 ns;
61
62        wait for clk_period*10;
63
64        Reset <= '1' ;
65        Din<= "10101010" ;
66
```

```
67        wait for 25 ns;
68
69        Load <= '1' ;
70
71        wait for 20 ns;
72
73
74        -- insert stimulus here
75
76        wait;
77     end process;
78
79  END;
80
```

خروجی شبیه سازی :



در کد شبیه سازی ابتدا ریست فعال می باشد و مقدار رجیستر برابر با ۰ می شود که در خروجی بعد یک سیکل آن را مشاهده می کنیم. سپس زمانی که ریست برداشته می شود و مقداری در ورودی قرار می گیرد تا زمانی که پایه لود فعال نشود داخل رجیستر ریخته نمی شود و خروجی همان صفر باقی می ماند. پس از فعال شدن پایه لود بعد یک سیکل خروجی تغییر می کند.

۲. واحد کانتر

کد برنامه :

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

entity Timer_8bits is
    Port ( Pre_Load_Count : in  STD_LOGIC_VECTOR (7 downto 0);
           UP_Down : in  STD_LOGIC;
           Pre_Load_Enable : in  STD_LOGIC;
           Pre_Reset_Enable : in  STD_LOGIC;
           RESET : in  STD_LOGIC;
           Clk : in  STD_LOGIC;
           Count : out  STD_LOGIC_VECTOR (7 downto 0);
           Interrupt : out  STD_LOGIC_VECTOR (0 downto 0) );
end Timer_8bits;

architecture Behavioral of Timer_8bits is

begin
process(clk)
variable Cunter_temp : integer range 0 to 255 := 0 ;
variable MIN_Counter : integer range 0 to 255 := 0 ;
variable MAX_Counter : integer range 0 to 255 := 255 ;
variable Interrupt_temp : integer range 0 to 1 := 0 ;
begin
if( clk'event and clk='1' ) then

   if Reset = '0' then
      Cunter_temp := 0 ;
      MIN_Counter := 0 ;
      MAX_Counter := 255 ;
   elsif Pre_Load_Enable = '1' then
      MIN_Counter := to_integer (unsigned(Pre_Load_Count)) ;
   elsif Pre_Reset_Enable = '1' then
      MAX_Counter := to_integer (unsigned(Pre_Load_Count)) ;
   end if ;
   ------------
   if UP_DOWN = '1' then
      if Cunter_temp > MIN_Counter-1 and Cunter_temp < MAX_Counter  then
      Cunter_temp := Cunter_temp + 1 ;
      Interrupt_temp :=0 ;
      else
      Cunter_temp := MIN_Counter ;
      Interrupt_temp :=1 ;
      end if ;
   elsif UP_DOWN = '0' then
      if Cunter_temp > MIN_Counter and Cunter_temp < MAX_Counter+1 then
      Cunter_temp := Cunter_temp - 1 ;
      Interrupt_temp :=0 ;
      else
      Cunter_temp := MAX_Counter ;
      Interrupt_temp :=1 ;
      end if ;
   end if ;
   ----------
   Count <= STD_LOGIC_VECTOR (to_unsigned(Cunter_temp,8)) ;
   Interrupt <= STD_LOGIC_VECTOR (to_unsigned(Interrupt_temp,1)) ;
end if ;
end process ;

end Behavioral;

```

در این برنامه کانتر از مقدار ۰ تا ۲۵۵ را می شمارد . ابتدا با دستورات شرطی در صورتی که مقدار بیشینه و کمینه کانتر تغییر کرده باشد توسط پایه های پری لود آن ها را به برنامه اعمال می کند . البته این تغییرات در سیکل بعدی بر روی کانتر تاثیر می گذارد و همچنان به

شمارش قبلی خود کانتر ادامه خواهد داد. در قسمت بعد نیز با توجه به سیگنال بالا رونده یا پایین رونده مقدار رجیستر را کاهش یا اضافه می نماید.

در آخر نیز یک شرط برای ایجاد اینتراپ نوشته شده است که زمانی که به ماکسیمم برسد و بخواهیم کانتر برنامه را به مقدار اولیه شمارش برگردانیم تغییر می کند و در سایر موارد برابر با صفر می باشد.

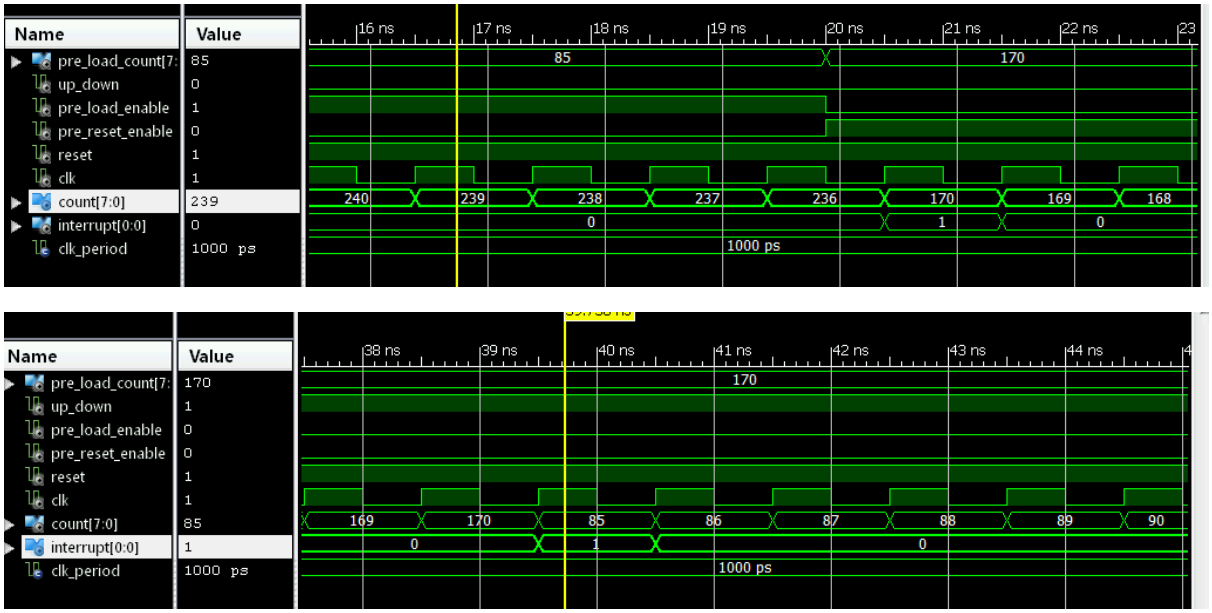
کد شبیه سازی :

```
1
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4  USE ieee.numeric_std.ALL;
5
6  ENTITY test IS
7  END test;
8
9  ARCHITECTURE behavior OF test IS
10
11     -- Component Declaration for the Unit Under Test (UUT)
12
13     COMPONENT Timer_8bits
14     PORT(
15          Pre_Load_Count : IN  std_logic_vector(7 downto 0);
16          UP_Down : IN  std_logic;
17          Pre_Load_Enable : IN  std_logic;
18          Pre_Reset_Enable : IN  std_logic;
19          RESET : IN  std_logic;
20          Clk : IN  std_logic;
21          Count : OUT  std_logic_vector(7 downto 0);
22          Interrupt : OUT  std_logic_vector(0 downto 0)
23         );
24     END COMPONENT;
25
26
27    --Inputs
28    signal Pre_Load_Count : std_logic_vector(7 downto 0) := (others => '0');
29    signal UP_Down : std_logic := '0';
30    signal Pre_Load_Enable : std_logic := '0';
31    signal Pre_Reset_Enable : std_logic := '0';
32    signal RESET : std_logic := '1';
33    signal Clk : std_logic := '0';
```

```
34
35    --Outputs
36    signal Count : std_logic_vector(7 downto 0);
37    signal Interrupt : std_logic_vector(0 downto 0);
38
39    -- Clock period definitions
40    constant Clk_period : time := 1 ns;
41
42  BEGIN
43
44    -- Instantiate the Unit Under Test (UUT)
45    uut: Timer_8bits PORT MAP (
46           Pre_Load_Count => Pre_Load_Count,
47           UP_Down => UP_Down,
48           Pre_Load_Enable => Pre_Load_Enable,
49           Pre_Reset_Enable => Pre_Reset_Enable,
50           RESET => RESET,
51           Clk => Clk,
52           Count => Count,
53           Interrupt => Interrupt
54         );
55
56    -- Clock process definitions
57    Clk_process :process
58    begin
59       Clk <= '0';
60       wait for Clk_period/2;
61       Clk <= '1';
62       wait for Clk_period/2;
63    end process;
64
65
66    -- Stimulus process
```

```
67      stim_proc: process
68      begin
69         -- hold reset state for 100 ns.
70         wait for 10 ns;
71
72         Pre_Load_Enable <= '1' ;
73         Pre_Load_Count <= "01010101" ;
74
75         wait for 10 ns;
76
77         Pre_Load_Enable <= '0' ;
78         Pre_Reset_Enable <= '1' ;
79         Pre_Load_Count <= "10101010" ;
80
81         wait for 10 ns;
82
83         Pre_Reset_Enable <= '0' ;
84         UP_DOWN <= '1' ;
85
86         wait for Clk_period*10;
87
88         -- insert stimulus here
89
90         wait;
91      end process;
92
93  END;
94
```

خروجی :



تایمر از ۲۵۵ شروع به شمارش به صورت پایین رونده می کند. سپس با مقدار پری لود ۸۵ و پری ریست ۱۷۰ بارگذاری می شود. در سیکل بعدی که پری ریست تغییر می کند مشاهده می کنیم که مقدار تایمر هم عوض می شود. همچنین اینتراپت را خواهیم داشت.