|  |
| --- |
|  |
| CA Lab – EX5 |
| |  |  |  | | --- | --- | --- | | MasoudHeidaryMH@gmail.com |  |  | |



برای ساخت رجیستر 8 بیتی به این صورت عمل میکنیم که یک variable 8 bit در داخل process تعریف کرده، سپس با توجه به کلاک بالارونده و وضعیت پایه ها در آن زمان، خروجی را تعیین میکنیم.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity Register8Bit is

    Port ( Din : in  STD\_LOGIC\_VECTOR (7 downto 0);

           Load : in  STD\_LOGIC;

           Reset : in  STD\_LOGIC;

           Clk : in  STD\_LOGIC;

           Dout : out  STD\_LOGIC\_VECTOR (7 downto 0));

end Register8Bit;

architecture Behavioral of Register8Bit is

-- signal reg: STD\_LOGIC\_VECTOR (7 downto 0) := "00000000";

begin

process (Clk)

    variable reg: STD\_LOGIC\_VECTOR (7 downto 0) := "00000000";

begin

    if (Clk'Event and Clk = '1') then

        if Reset = '1' then

            reg := "00000000";

        elsif Load = '1' then

            reg := Din;

        end if;

    end if;

    Dout <= reg;

end process;

end Behavioral;

test bench:

تست این ماژول به راحتی طراحی آن است و به صورت زیر عمل میکنیم

\* قابل توجه است به کدهای auto generate شده دست نمیزنیم \*

stim\_proc: process

   begin

      Load <= '0';

      Reset <= '0';

      wait for Clk\_period;

      Din <= "11110000";

      Load <= '1';

      wait for Clk\_period;

      Load <= '0';

      wait for Clk\_period;

      Reset <= '1';

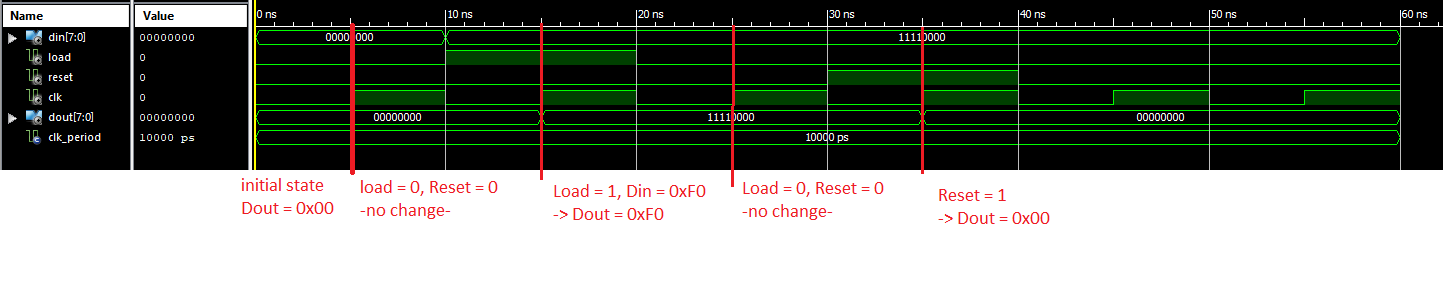
      wait for Clk\_period;

      Reset <= '0';

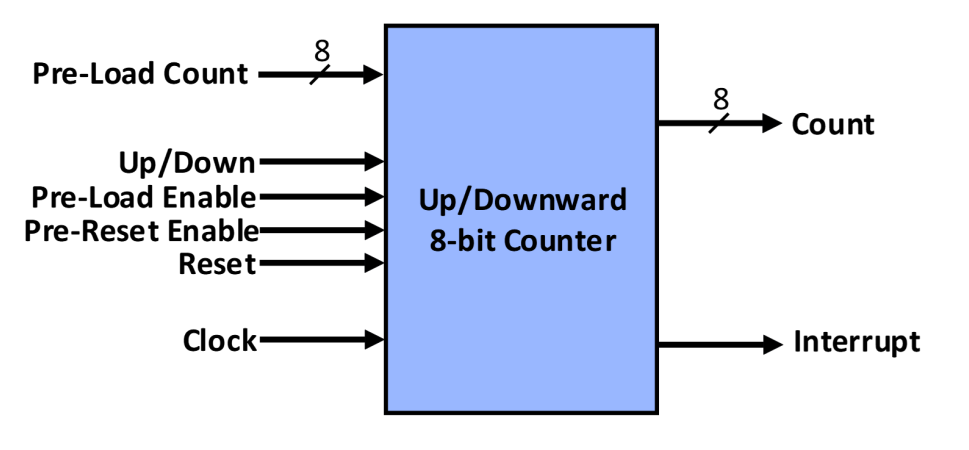
      wait for Clk\_period;

      wait;

   end process;

output:

طراحی counter



ایده ها:

برای شمارش، ابتدا یک متغیر تعریف میکنیم

اگر در حالت بالارونده باشیم، با آمدن هر کلاک، به آن یک واحد اضافه میکنیم

اگر PreResetEn = 1 باشد، با رسیدن متغیر به عدد ورودی، متغیر را 0 میکنیم

اگر PreResetEn = 0 باشد، هر زمانی که متغیر بزرگتر از 255 شد، باید آن را ریست کنیم، که خود دو حالت دارد

اگر PreLoadEn = 0 باشد، متغیر را 0 خواهیم کرد، در غیر این صورت برابر عدد ورودی قرار خواهیم داد.

برای قسمت پایین رونده نیز ایده به صورتی ثابت است،

با آمدن هر کلاک، متغیر را یک واحد کوچکتر میکنیم،

اگر PreResetEn = 1 باشد، با رسیدن متغیر به عدد ورودی، متغیر را به 255 ریست میکنیم

اگر PreResetEn = 0 باشد، هر زمانی که متغیر کوچکتر از 0 شد باید آن را ریست کنیم، که خود دو حالت دارد

اگر PreLoadEn = 0، متغیر را 255 خواهیم کرد، در غیر این صورت برابر عدد ورودی قرار خواهیم داد

برای قسمت های Reset, Interrupt، ایده ها ساده هستند، و از توضیح آنها صرف نظر میکنم.

Code

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity Counter is

    Port ( PreInput : in  STD\_LOGIC\_VECTOR (7 downto 0);

           Count : out  STD\_LOGIC\_VECTOR (7 downto 0);

           UpDown : in  STD\_LOGIC;

           PreLoadEn : in  STD\_LOGIC;

           PreResetEn : in  STD\_LOGIC;

           Reset: in STD\_LOGIC;

           Clk : in  STD\_LOGIC;

           Interrupt : out  STD\_LOGIC);

end Counter;

architecture Behavioral of Counter is

begin

process (Clk, Reset)

    variable reg: integer := 0;

begin

    if Reset = '1' then

        if PreLoadEn = '1' then

            reg := to\_integer(unsigned(PreInput));

        elsif UpDown = '0' then

            reg := 0;

        else

            reg := 255;

        end if;

    end if;

    -- instead of \_\_\_ if (Clk'Event and Clk = '1') then

    if rising\_edge(Clk) and Reset = '0' then

        Interrupt <= '0';

        -- up routine

        if UpDown = '0' then

            reg := reg + 1;

            -- natural reset

            if reg > 255 then

                if PreLoadEn = '1' then

                    reg := to\_integer(unsigned(PreInput));

                else

                    reg := 0;

                end if ;

                Interrupt <= '1';

            -- pre reset en

            elsif PreResetEn = '1' then

                if reg = to\_integer(unsigned(PreInput)) then

                    reg := 0;

                    Interrupt <= '1';

                end if ;

            end if ;

        -- down routine

        else

            reg := reg - 1;

            -- natural reset

            if reg < 0 then

                if PreLoadEn = '1' then

                    reg := to\_integer(unsigned(PreInput));

                else

                    reg := 255;

                end if ;

                Interrupt <= '1';

            -- pre reset en

            elsif PreResetEn = '1' then

                if reg = to\_integer(unsigned(PreInput)) then

                    reg := 255;

                    Interrupt <= '1';

                end if ;

            end if ;

        end if;

    end if ;

    -- set output

    Count <= std\_logic\_vector(to\_unsigned(reg, 8));

end process;

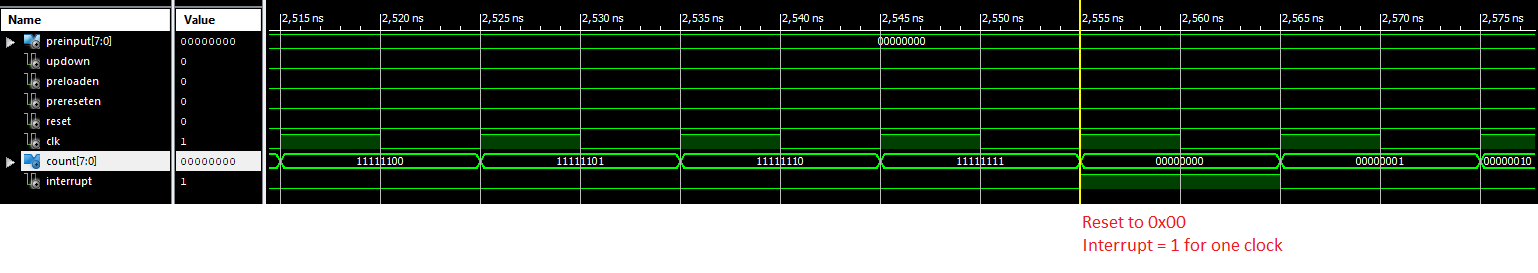
end Behavioral;

Test Bench:

ابتدا ماژول را در حالت های مختلف برای بالارونده چک میکنیم

Normal:

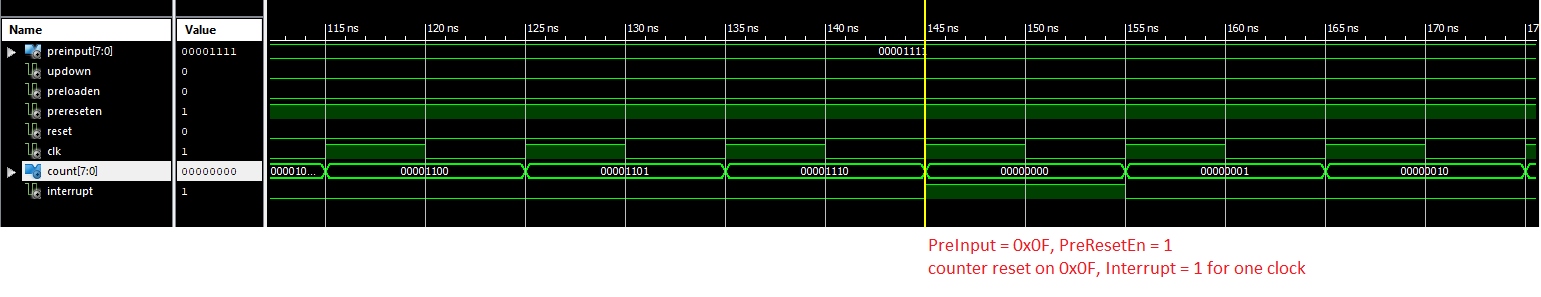
      UpDown <= '0';



PreResetEn:

UpDown <= '0';

      PreInput <= "00001111";

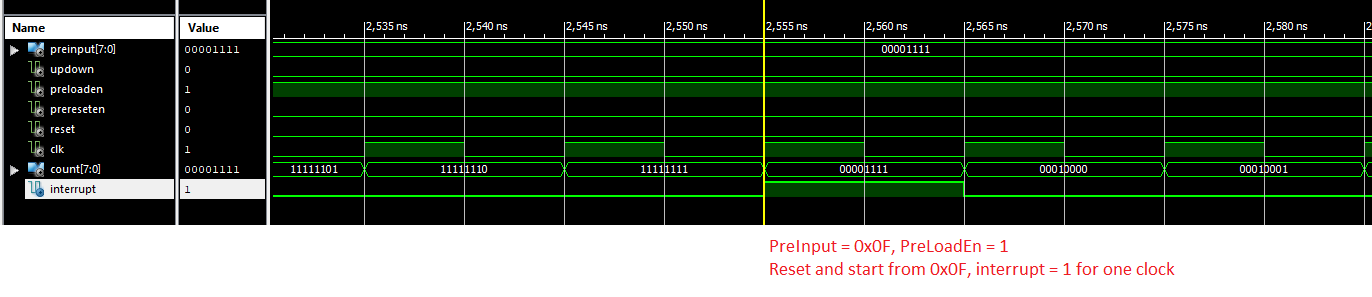
      PreResetEn <= '1';

PreLoadEn:

      UpDown <= '0';

      PreInput <= "00001111";

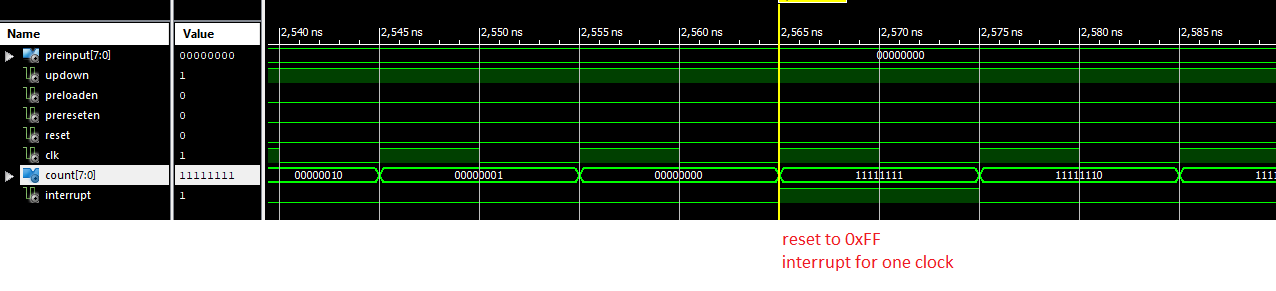
      PreLoadEn <= '1';



حال ماژول را به صورت پایین رونده نیز تست میکنیم

Normal:

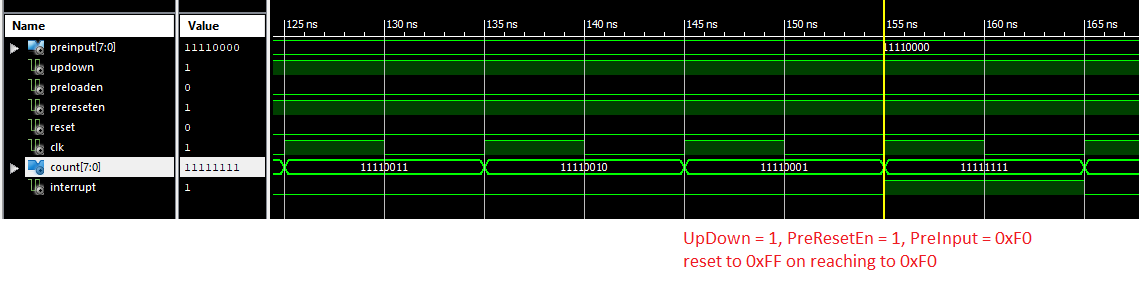
      UpDown <= '1';



PreResetEn:

      UpDown <= '1';

      PreInput <= "11110000";

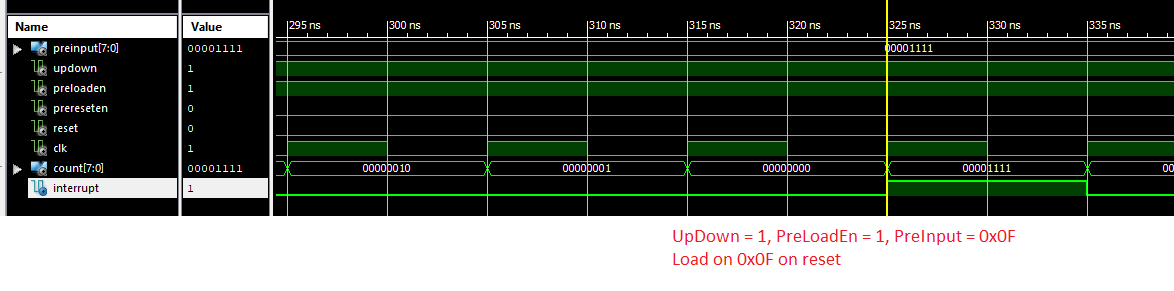
      PreResetEn <= '1';

PreLoadEn:

      UpDown <= '1';

      PreInput <= "00001111";

      PreLoadEn <= '1';



حال که تمامی حالت ها برای قسمت های بالارونده و پایین رونده تست شدند، باید پایه Reset را نیز تست کنیم، دقت کنید که پایه Reset به صورت aSync طراحی شده است.

Upward, normal

      UpDown <= '0';

      wait for Clk\_period;

      wait for Clk\_period;

      wait for Clk\_period;

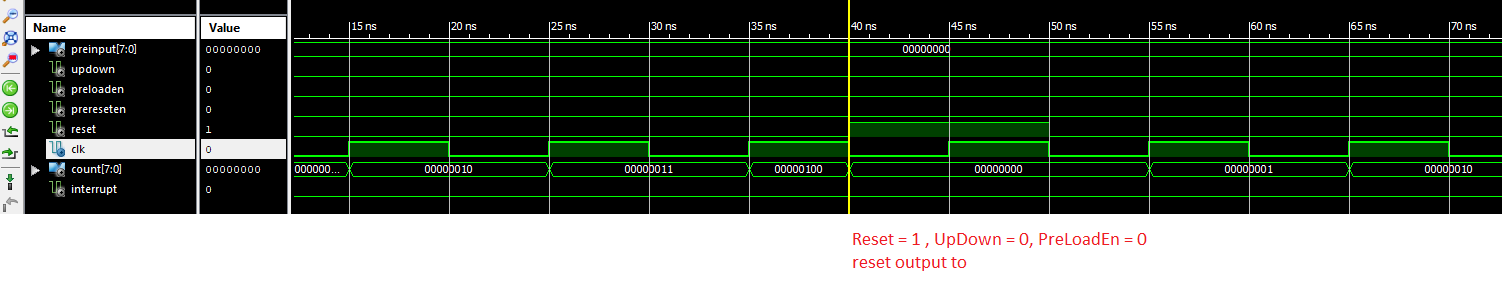
      wait for Clk\_period;

      Reset <= '1';

      wait for Clk\_period;

      Reset <= '0';

      wait for Clk\_period;



Downward, normal

      UpDown <= '1';

      wait for Clk\_period;

      wait for Clk\_period;

      wait for Clk\_period;

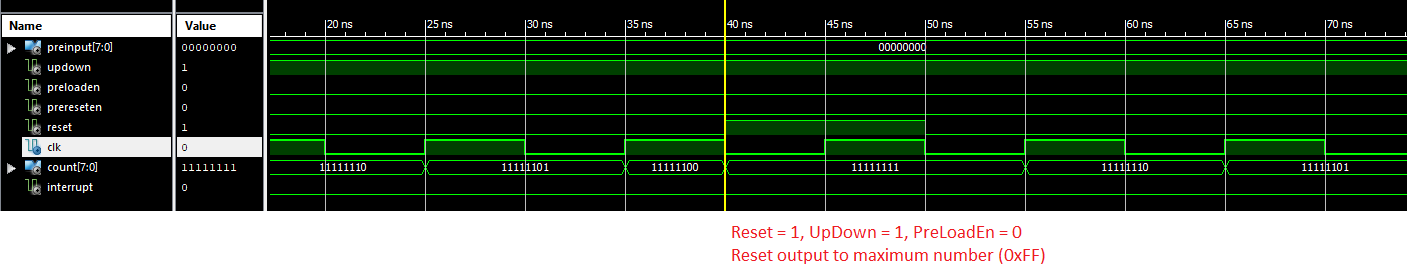
      wait for Clk\_period;

      Reset <= '1';

      wait for Clk\_period;

      Reset <= '0';

      wait for Clk\_period;



PreLoadEn (upward OR downward)

      UpDown <= '0';

      PreInput <= "00001111";

      PreLoadEn <= '1';

      wait for Clk\_period;

      wait for Clk\_period;

      wait for Clk\_period;

      wait for Clk\_period;

      Reset <= '1';

      wait for Clk\_period;

      Reset <= '0';

      wait for Clk\_period;

