پارسا محمدپور – ۹۸۲۴۳۰۵۰

پویا جهانگیری – ۹۸۲۴۳۰۷۶

گزارش جلسه:

در این جلسه ابتدا به پیاده‌سازی ماژول ضرب‌کننده دو عدد n بیتی رفتیم اما در این قسمت به مشکل برخوردیم و تمام مدت این جلسه و حتی تایمی هم در خانه مشغول پیاده‌سازی آن بودیم. اما سپس به سرانجام رسید. در ادامه توضیح کدهای زده شده به همراه شکل خروجی در مدل‌سیم آورده شده است.

توضیح کد. کدهای زده شده برای این قسمت به صورت زیر می‌باشد:

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.numeric_std.all;

entity multiplier is
generic (N: integer := 4);
port (
        clk    : in std_logic;
        a      :  in  std_logic_vector(N-1 downto 0);
        b      :  in  std_logic_vector(N-1 downto 0);
        res :  out std_logic_vector(2*N-1 downto 0) );
end entity multiplier;
```

در این قسمت تنها کدهایی زده شده است که مربوط به import کردن کتابخانه‌های مورد استفاده و تعریف کردن ماژول ورودی می‌باشد. برای این کار همانظور که در صورت سوال قید شده است، برای دریافت نمره امتیازی، این ماژول را به صورت جنریک (generic) پیاده‌سازیکردیم که برای آن، ورودی N را در نظر گرفته‌اسم تا برنامه با آن کار کند.

```
architecture behavioral of multiplier is

begin

process(clk)
variable b_extended, b_num : STD_LOGIC_VECTOR(2*N-1 downto 0) := (others => '0');
variable counter : INTEGER := 0;
variable pre_a, pre_b : STD_LOGIC_VECTOR(N-1 downto 0);
```

د راین قسمت که مشاهده می‌کنیم، تیکه کدهای مربوط به تعریف متغییرها و مقدار دهی اولیه آن‌ها قرار گرفته است. این کار به صورت نشان داده شده در تصویر بالا می‌باشد.

```
begin
    if rising_edge(Clk) then
        if counter = 0 then
            b_num(2*N-1 downto N) := (others => '0');
            b_num(N-1 downto 0) := b;
            b_extended := (others => '0');
            pre_a := a;
            pre_b := b;
        end if;
        if counter < N then
            if a(counter) = '1' then
                b_extended := std_logic_vector(unsigned(b_extended) + unsigned(b_num));
            end if;
            b_num := std_logic_vector(shift_left(unsigned(b_num), 1));
        end if;
        counter := counter + 1;
        if counter > N-1 then
            if not (pre_a = a and pre_b = b) then
                counter := 0;
            end if;
        end if;
        res <= b_extended;
    end if;
    --b_num <= "0000" & b;
    --b_extended <= "1010" & a;
    --for i in 0 to N-1 loop
    --  if a(i) = '1' then
    --      b_extended <= std_logic_vector(unsigned(b_extended) + unsigned(b_num));
    --  end if;
    --  b_num <= std_logic_vector(shift_left(unsigned(b_num), 1));
    --  wait for 2 ns;
    --end loop;
    --res <= b_extended;
end process;
end behavioral;
```

سپس در این قسمت کدهای مربوط به پیاده‌سازی منطق این ماژول ضرب‌کننده رفتیم. در این ماژول، در هر سری، متغییر داخلی از صفر تا N حرکت می‌کند و به ازای هر عدد در درون این بازه، ابتدا در صورت یک بودن بیت ورودی اول، متغییر کنونی را با متغییر دوم جمع می‌کنیم. اما در این کار یک مشکلی وجود داشت که در ادامه به شرح آن و راه حل اعمال شده می‌پردازیم.

در این کد، همه چیز مطابق توضیحات ارائه شده می‌باشد به جز آخرین if ای که در این کد قرار گرفته است. دلیل وجود این if چیست؟ اگر در این کد این if آخر را قرار نمی‌دادیم، آن‌وقت همینطور تا آخر مدام عملیات ضرب تکرار می‌شد و همینطور به از اول شروع کردن عملیات ادامه می‌داد. برای جلوگیری از این اتفاق، این if را اضافه کردیم تا از این امر جلوگیری کنیم.

در ادامه کدهای مربوط به فایل testBench را قرار می‌دهیم:

```
library ieee;
use ieee.std_logic_1164.all;

entity multiplier_tb is end entity;

architecture behavior of multiplier_tb is
 COMPONENT multiplier
 GENERIC (N      : INTEGER := 4);
 PORT(
    clk    : in std_logic;
    a,b : in STD_LOGIC_VECTOR(3 downto 0);
    res: out STD_LOGIC_VECTOR(7 downto 0)
 );
 END COMPONENT;

    signal clk    : std_logic := '1';
    signal a, b         : std_logic_vector(3 downto 0);
    signal res      : std_logic_vector(7 downto 0);
begin
uut: multiplier
        generic map (
            N => a'length
        )
        port map (
            clk => clk,
            a     => a,
            b      => b,
            res => res
        );
```

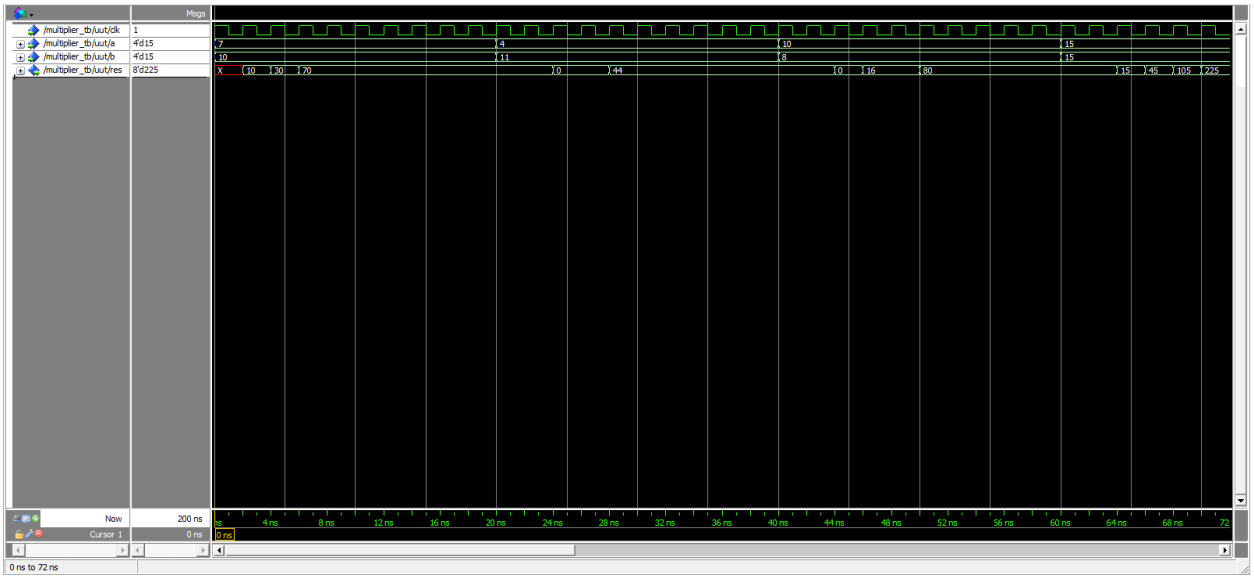
```
-- Process for generating the clock
    clk <= not clk after 1 ns;

    stim_proc: process
    begin
        a <= "0111";
        b <= "1010";
        wait for 20 ns;
        a <= "0100";
        b <= "1011";
        wait for 20 ns;
        a <= "1010";
        b <= "1000";
        wait for 20 ns;
        a <= "1111";
        b <= "1111";
        wait;
    end process;
end architecture;
```

در این تستبنچ تنها نکته جدید وجود کلاک است که پیاده‌سازی آن مطابق با کد همین قسمت آورده شده می‌باشد.

شبیه‌سازی:

در ادامه عکس حاصل از شبیه‌سازی این کد را در نرم‌افزار مدل‌سیم قرار می‌دهیم:


Msgs
/multiplier_tb/uut/clk
/multiplier_tb/uut/a
/multiplier_tb/uut/b
/multiplier_tb/uut/res
1
4'd15
4'd15
8'd225
7
4
10
15
10
11
8
15
X
10
30
70
0
44
0
16
80
15
45
105
225
Now
200 ns
Cursor 1
0 ns
0 ns
4 ns
8 ns
12 ns
16 ns
20 ns
24 ns
28 ns
32 ns
36 ns
40 ns
44 ns
48 ns
52 ns
56 ns
60 ns
64 ns
68 ns
72
0 ns to 72 ns