به نام خدا

پروژه ی درس CAD

دکتر خدادادی

اعضای گروه:

علیرضا اسلامی خواه(۹۹۵۲۱۰۶۴)

امیرمحمد کمیجانی(۹۹۵۲۲۰۳۲)

آرین عبدالهی ثابت نژاد(۹۹۵۲۱۴۴۲)

امیرعلی فرازمند(۹۹۵۲۲۳۲۹)

موضوع پروژه:

RAM تک پورتی ۱۲۸ بایتی

با این قابلیت که خانه ها هرکدام در زمان مشخصی که از ورودی میگیرند میتوانند طوری تنظیم شوند که پاک شوند یا پاک نشوند

در این پروژه ما یک رم تک پورتی طراحی کردیم با حافظه ی ۱۲۸ بایتی(۸x۱۲۸ بیتی).پورت انتیتی رم مان از این قرار است:

```vhdl
entity Single_port_RAM_VHDL is
    port (
        RAM_ADDR: in std_logic_vector(6 downto 0);  -- Address to write/read RAM
        RAM_DATA_IN: in std_logic_vector(7 downto 0);  -- Data to write into RAM
        RAM_WR: in std_logic;  -- Write enable
        RAM_CLOCK: in std_logic;  -- Clock input for RAM
        RAM_DATA_OUT: out std_logic_vector(7 downto 0);  -- Data output of RAM
        timer: in unsigned(7 downto 0);  -- Timer value for the address
        infinityFlag: in std_logic  -- Infinity flag for the address
    );
end Single_port_RAM_VHDL;
```

- **RAM_ADDR** پورت ورودی که آدرس نوشتن یا خواندن از RAM را نمایش می‌دهد(در این آدرس دیتا باید نوشته شود).
- **RAM_DATA_IN** پورت ورودی که دیتا برای نوشتن در RAM میرود.
- **RAM_WR** پورت ورودی که سیگنال فعال سازی نوشتن برای RAM را نمایش می‌دهد. (write enable)
- **RAM_CLOCK** پورت ورودی که سیگنال کلاک برای RAM را نمایش می‌دهد.
- **RAM_DATA_OUT** پورت خروجی که داده خروجی از RAM را نمایش می‌دهد.
- **Timer** پورت ورودی که مقدار تایمر برای آدرس را نمایش می‌دهد (درصورتی که infinityFlag ما ۰ باشد از آن کاسته میشود).
- **infinityFlag** پورت ورودی std_logic که برای آدرس میگوید که آیا مقدار خانه باید تا بینهایت بماند یا که با کم شدن تایمر و در نهایت با به ۰ رسیدن آن دیتا را از رم پاک کند.

برای ۳بخش گفته شده برای هر خانه در architectureمان تایپ های متناظری باید تعریف کنیم:

```vhdl
architecture Behavioral of Single_port_RAM_VHDL is
    type RAM_ARRAY is array (0 to 127) of std_logic_vector(7 downto 0);
    type TIMER_ARRAY is array (0 to 127) of unsigned(7 downto 0);
    type INFINITY_ARRAY is array (0 to 127) of std_logic;
```

- **RAM_ARRAY** یک نوع آرایه که حافظه RAM را با ۱۲۸ عنصر نشان می‌دهد، هر عنصر یک std_logic_vector ۸ بیتی است.

- **TIMER_ARRAY** آرایه ای از unsigned که تایمرها را برای هریک از ۱۲۸ عنصر ذخیره میکند.
- **INFINITY_ARRAY** آرایه که با ۰ و ۱ برای ۱۲۸ عنصر مشخص میکند که آیا با شروع به تمام شدن تایمرش کند و با به ۰ رسیدن آن نابود شود یا نه.

در ادامه سیگنال هایی از تایپ های گفته شده میسازیم و تمام بیت های آن ها را با ۰ در ابتدا مقداردهی میکنیم:

```
21
22  signal RAM: RAM_ARRAY :=(
23     x"55",x"66",x"77",x"67",-- 0x00:
24     x"99",x"00",x"00",x"11",-- 0x04:
25     x"00",x"00",x"00",x"00",-- 0x08:
54     x"00",x"00",x"00",x"00"
55     );
56      signal timers: TIMER_ARRAY := (
57          (others => (others => '0'))  -- Initialize all timers to zero
58      );
59      signal infinityFlags: INFINITY_ARRAY := (
60          (others => '0')  -- Initialize all infinity flags to zero
61      );
```

حال به بخش اصلی قطعه مان میرسیم، جایی که منطقش را تعریف کرده ایم:

```
62
63  begin
64      process (RAM_CLOCK)
65      begin
66          if rising_edge(RAM_CLOCK) then
67              if RAM_WR = '1' then  --write enable
68                  RAM(to_integer(unsigned(RAM_ADDR))) <= RAM_DATA_IN;
69                  timers(to_integer(unsigned(RAM_ADDR))) <= timer;
70                  infinityFlags(to_integer(unsigned(RAM_ADDR))) <= infinityFlag;
71              end if;
72
73              for i in timers'range loop
74                  if timers(i) > 0 and infinityFlags(i) = '0' then
75                      timers(i) <= timers(i) - 1;
76                  elsif timers(i) = 0 then
77                      RAM(i) <= (others => '0');  -- Reset data when timer reaches zero
78                  end if;
79              end loop;
80          end if;
81      end process;
82
83      RAM_DATA_OUT <= RAM(to_integer(unsigned(RAM_ADDR)));
84
85  end Behavioral;
```

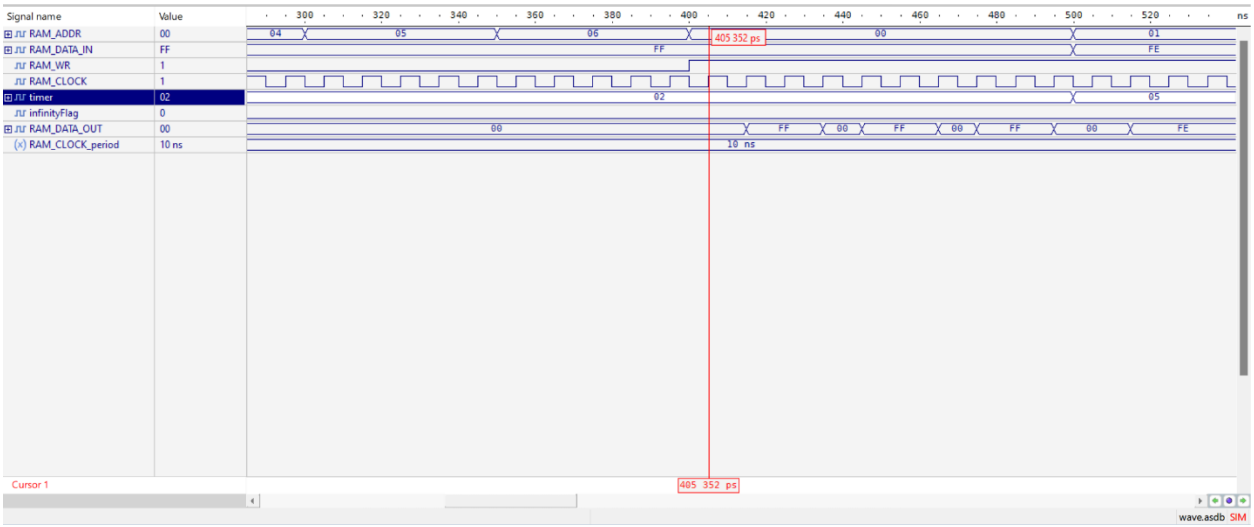
در هر کلاک بالارونده قطعه در خط ۶۷ تا ۷۰ چک میکند که اگر دیتای جدیدی باید وارد شود(با RAM_WR)، آنرا و تایمر و infinityFlag مربوطه اش وارد رم میکند.

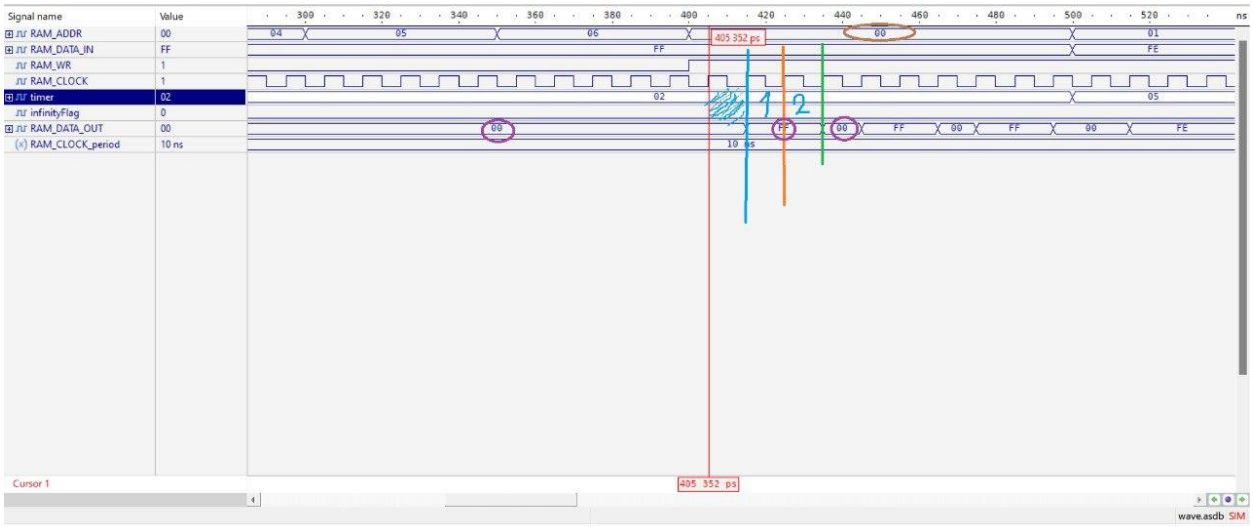
در خط ۷۳ تا ۷۹ هم برای ۱۲۸خانه چک میکند که اگر infinityFlagشان ۰ بود و تایمر هنوز به ۰ نرسیده بود، از تایمر متناظر آن ها ۱ واحد کم کند و اگر تایمر خانه ای ۰ شد باید خانه ی آن را در رم ۰ کند در RAM مان دیگر جای آن دیتا "۰۰۰۰۰۰۰۰" خواهیم داشت، infinityFlagش تغیری نمیکند و تایمرش هم قاعتا به ۰ رسیده و آن را هم تغیری نمیدهیم،

چرا که نیازی نیست. همچنین در حالتی که infinityFlag خانه ای ۱ بود کلا با آن سر این قضیه کاری نداریم.

نمونه از تستی که انجام داده ایم با کد مربوط بهش:

```
    RAM_CLOCK_process : process
    begin
        RAM_CLOCK <= '0';
        wait for RAM_CLOCK_period/2;
        RAM_CLOCK <= '1';
        wait for RAM_CLOCK_period/2;
    end process;

    stim_proc: process
    begin
        RAM_WR <= '0';
        RAM_ADDR <= "0000000";
        RAM_DATA_IN <= x"FF";
        timer <= to_unsigned(2, 8);  -- Set timer value
        infinityFlag <= '0';  -- Set infinity flag to false

    -- Start reading data from RAM
        for i in 0 to 5 loop
            RAM_ADDR <= std_logic_vector(unsigned(RAM_ADDR) + 1);
            wait for RAM_CLOCK_period * 5;
        end loop;

        RAM_ADDR <= "0000000";
        RAM_WR <= '1';

         wait for 50 ns;
         infinityFlag <= '1';  -- Set infinity flag to true
         wait for 30 ns;
         infinityFlag <= '0';  -- Set infinity flag to true
         wait for 20 ns;
         infinityFlag <= '0';  -- Set infinity flag to true
        -- Start writing to RAM
        wait for 100 ns;

        for i in 0 to 5 loop
            RAM_ADDR <= std_logic_vector(unsigned(RAM_ADDR) + 1);
            RAM_DATA_IN <= std_logic_vector(unsigned(RAM_DATA_IN) - 1);
            timer <= to_unsigned(5, 8);  -- Set timer value
            infinityFlag <= '0';  -- Set infinity flag to false
            wait for RAM_CLOCK_period * 5;
        end loop;
        RAM_WR <= '0';
        wait;
    end process;
END;
```





در این عکس که wave form ماست اومدیم تایمر رو برابر ۲ قرار دادیم و بعدش از زمانی که FF رو به یک خونه از رم assign میکنیم دو تا کلاک میگذرد و آن را ریست میکند.