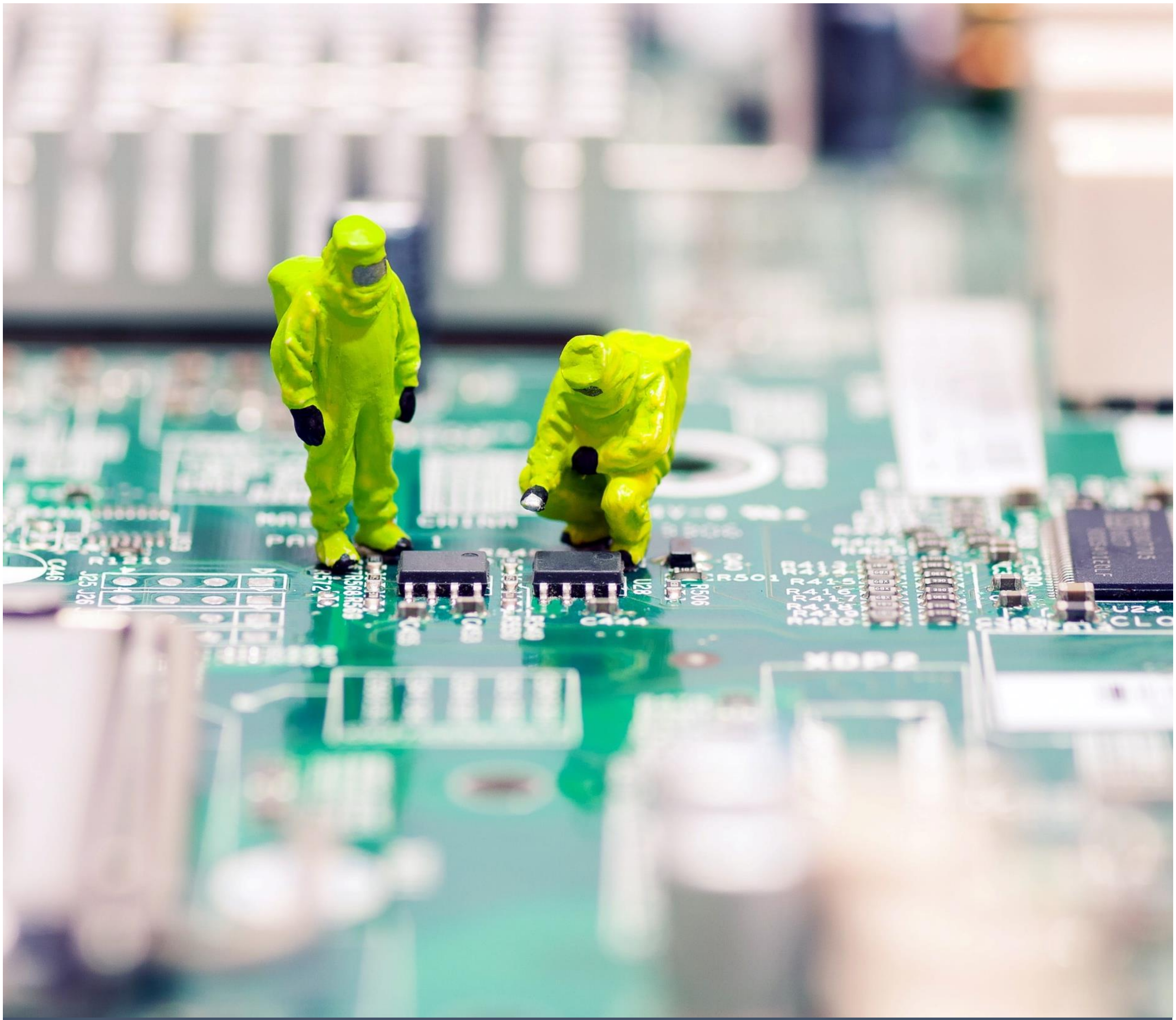


MH

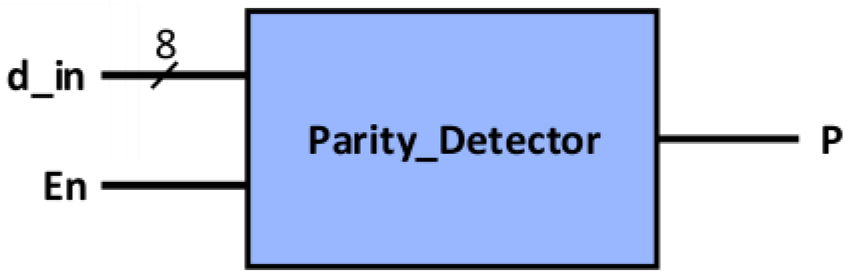
Masoud Heidary

think outside the box

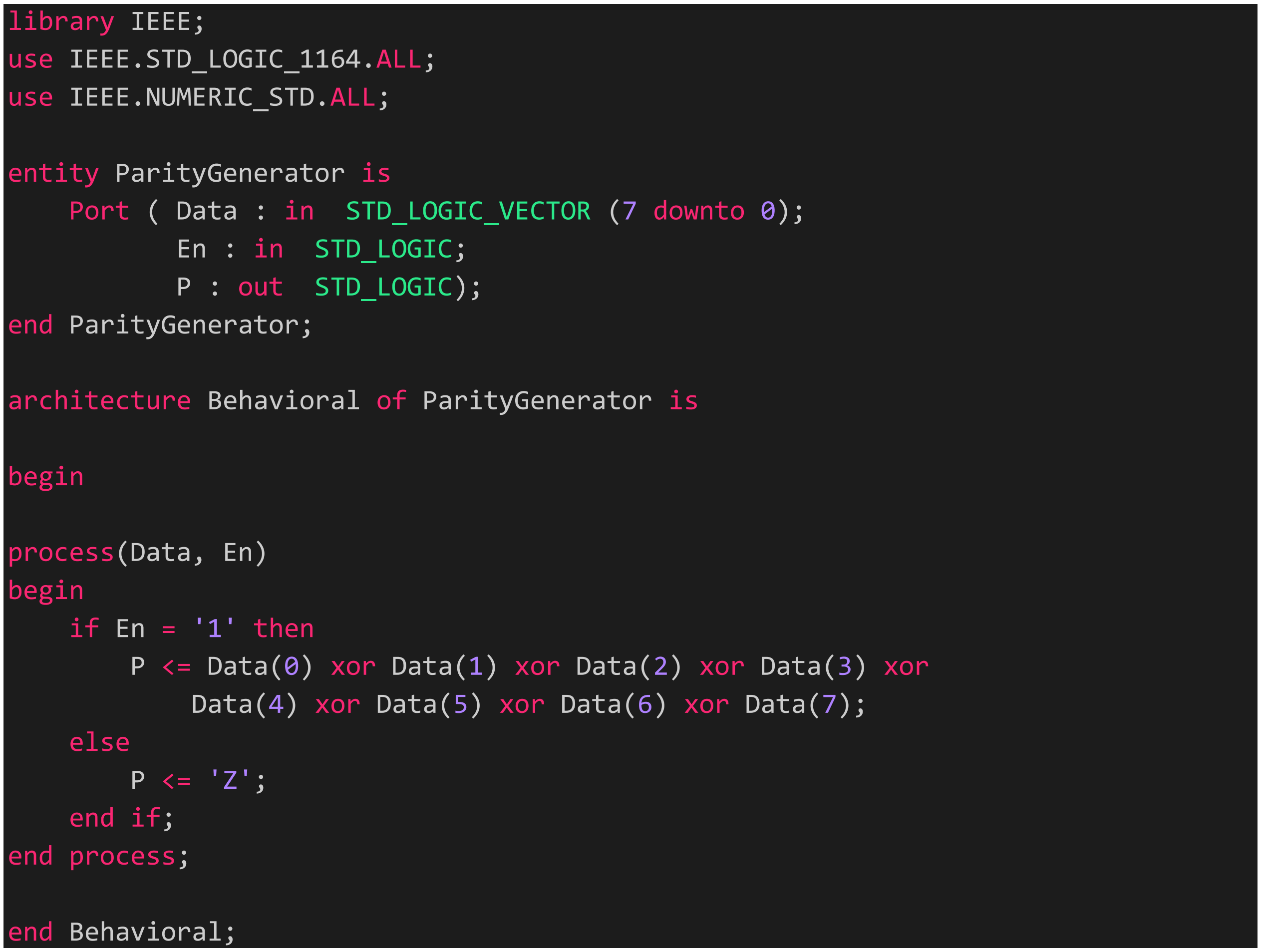
CA Lab – EX4

MasoudHeidaryMH@gmail.com

برای ساخت even parity generator میدانیم با xor کردن ورودی ها میتوانیم خروجی parity bit را بدست آوریم.

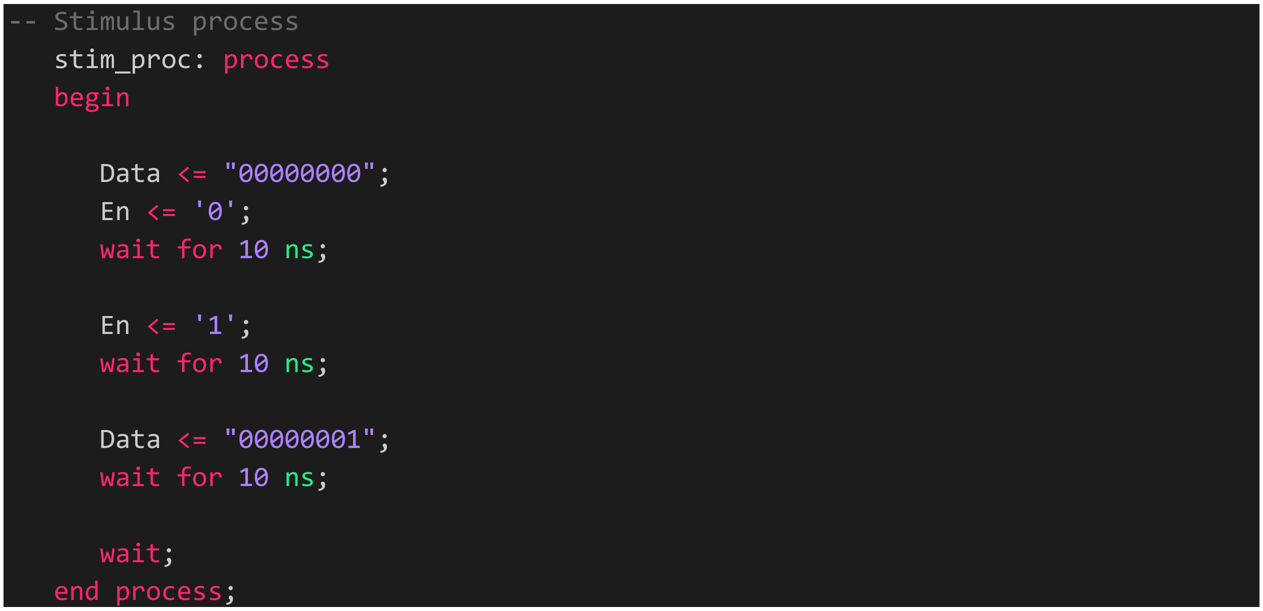


به این صورت کد را به صورت زیر داریم



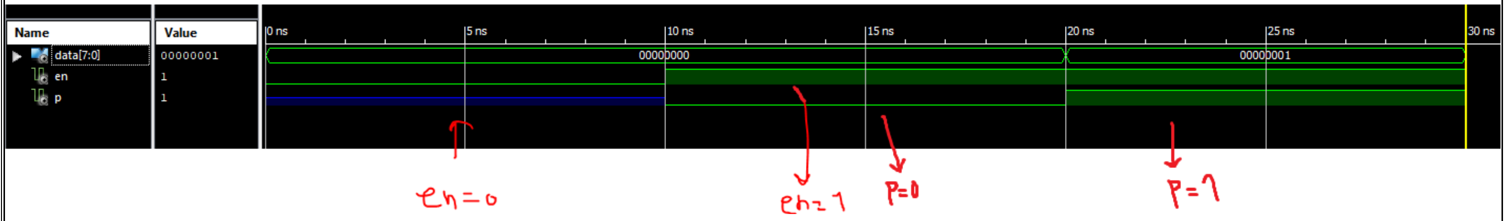
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity ParityGenerator is
    Port ( Data : in  STD_LOGIC_VECTOR (7 downto 0);
           En : in  STD_LOGIC;
           P : out  STD_LOGIC);
end ParityGenerator;

architecture Behavioral of ParityGenerator is

begin

process(Data, En)
begin
    if En = '1' then
        P <= Data(0) xor Data(1) xor Data(2) xor Data(3) xor
            Data(4) xor Data(5) xor Data(6) xor Data(7);
    else
        P <= 'Z';
    end if;
end process;

end Behavioral;
```

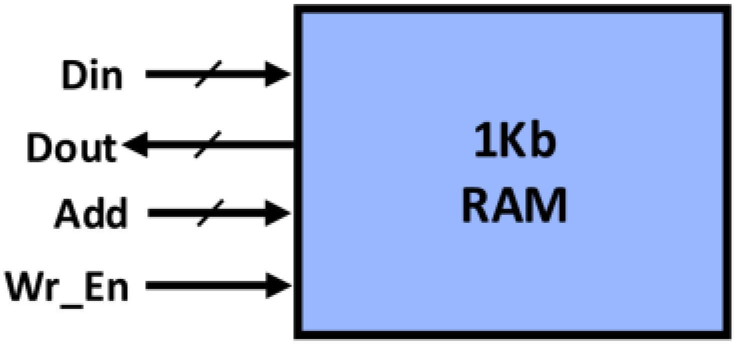
test bench:



```
-- Stimulus process
   stim_proc: process
   begin

      Data <= "00000000";
      En <= '0';
      wait for 10 ns;

      En <= '1';
      wait for 10 ns;

      Data <= "00000001";
      wait for 10 ns;

      wait;
   end process;
```

output:





memory به سایز 1kbit، در واقع 128-byte است، که به 7 خط آدرس نیاز دارد

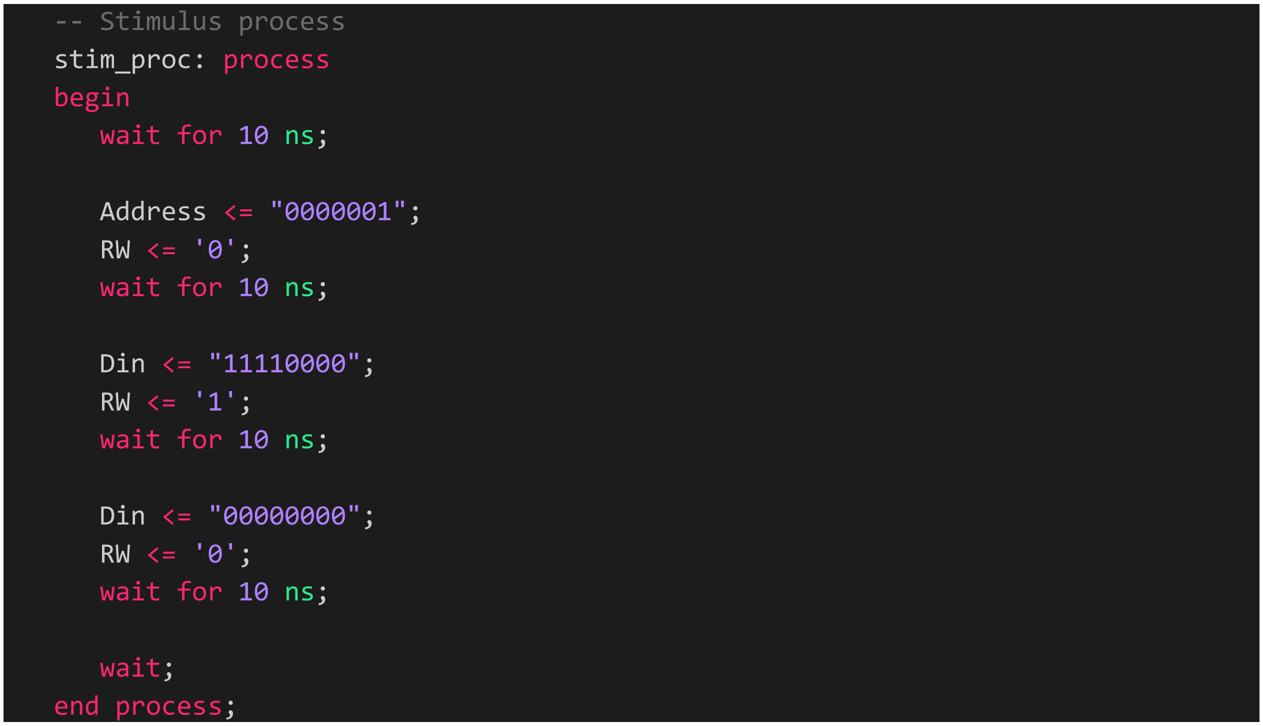
برای ساخت خانه های حافظه نیز به صورت عمل میکنیم

```
type RAM_128B is array (0 to 127) of STD_LOGIC_VECTOR (7 downto 0);
    signal RAM: RAM_128B := (others => (others => '0'));
```

و کد نویسی به صورت زیر میشود

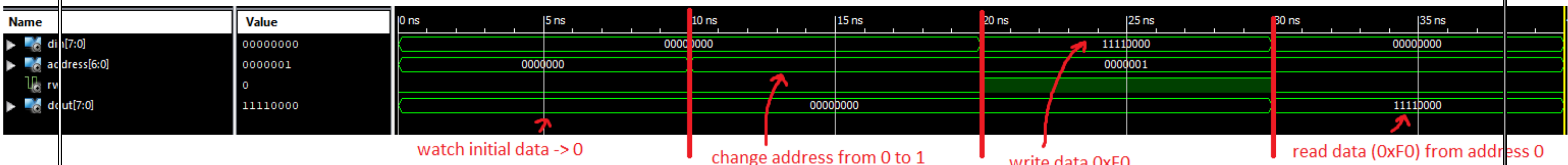
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Memory128Byte is
    Port ( Din : in  STD_LOGIC_VECTOR (7 downto 0);
           Dout : out  STD_LOGIC_VECTOR (7 downto 0);
           Address : in  STD_LOGIC_VECTOR (6 downto 0);
           RW : in  STD_LOGIC);
end Memory128Byte;

architecture Behavioral of Memory128Byte is
    type RAM_128B is array (0 to 127) of STD_LOGIC_VECTOR (7 downto 0);
    signal RAM: RAM_128B := (others => (others => '0'));
begin

process (Din, Address, RW)
begin
    if RW = '0' then
        Dout <= RAM(to_integer(unsigned(Address)));

    elsif RW = '1' then
        RAM(to_integer(unsigned(Address))) <= Din;

    end if ;
end process;

end Behavioral;
```

test bench:



```
-- Stimulus process
stim_proc: process
begin
   wait for 10 ns;

   Address <= "0000001";
   RW <= '0';
   wait for 10 ns;

   Din <= "11110000";
   RW <= '1';
   wait for 10 ns;

   Din <= "00000000";
   RW <= '0';
   wait for 10 ns;

   wait;
end process;
```

output:



watch initial data -> 0

change address from 0 to 1

write data 0xF0

read data (0xF0) from address 0
(input: 0)