Computer Architecture Lab Session 3

# ROM & RAM

# Bardia Ardakanian 9831072

# Ali Asad 9831004

## طراحی یک Read-only memory با زبان VHDL

برای نوشتن یک Read-only memory ابتدا موجودیت Rom را تعریف می‌کنیم، برای سنکروم بودن ROM برایش clock و reset می‌گذاریم. در این برنامه ما سعی در طراحی یک ROM با اندازه 16x8 بوده‌ایم. به حالتی که اعداد 0 الی 15 را در خود ذخیره کند. کد زیر برنامه موجودیت ROM است.

|  |
| --- |
| library ieee; use ieee.std\_logic\_1164.all; use ieee.std\_logic\_arith.all; use ieee.std\_logic\_unsigned.all;  --ROM with clock and reset --each word is 1 bytes (or 8 bits) --this architecture is 16 x 8 entity ROM is port(  clk : in std\_logic;  rst : in std\_logic;  enable : in std\_logic;  read : in std\_logic;  address : in integer range 0 to 15;  dout : out std\_ulogic\_vector(7 downto 0) ); end ROM;  architecture behave of ROM is  type ROM\_Array is array (0 to 15) of std\_ulogic\_vector (7 downto 0);   constant content: ROM\_Array := (  "00000000", "00000001", "00000010", "00000011",  "00000100", "00000101", "00000110", "00000111",  "00001000", "00001001", "00001010", "00001011",  "00001100", "00001101", "00001110", "00001111"  );  begin      process(clk, rst, enable, read, address)     begin         if(rst = '1') then              dout <= "ZZZZZZZZ";         elsif(clk'event and clk = '1') then             if (enable = '1') then                 if (read = '1') then                      dout <= content(address);                 else                      dout <= "ZZZZZZZZ";                 end if;             end if;         end if;     end process; end behave; |

## Test Bench برای ROM

برای طراحی Test Bench ابتدا باید موجودیت ROM\_TB را بسازیم، حال سیگنال های ورودی را نعریف می‌کنیم. در ابتدا سیگنال clock را 1 می‌گذاریم و هر 5 نانو ثانیه ان را not می‌کنیم تا ROM ما سنکروم شود.

بعد از این مرحله سیگنال reset را 1 می‌گذاریم تا برنامه از حالت خاصی شروع نشود. مقدار اولیه سیگنال address را 0 می‌گذاریم زیرا می‌خواهیم از خانه 0 الی 15 را بخوانیم.

بعد از 20 نانو ثانیه تاخیر سیگنال reset را 0 می‌کنیم و سیگنال read را هم 1 می‌کنیم تا امکان خواندن داده از ROM فراهم شود.

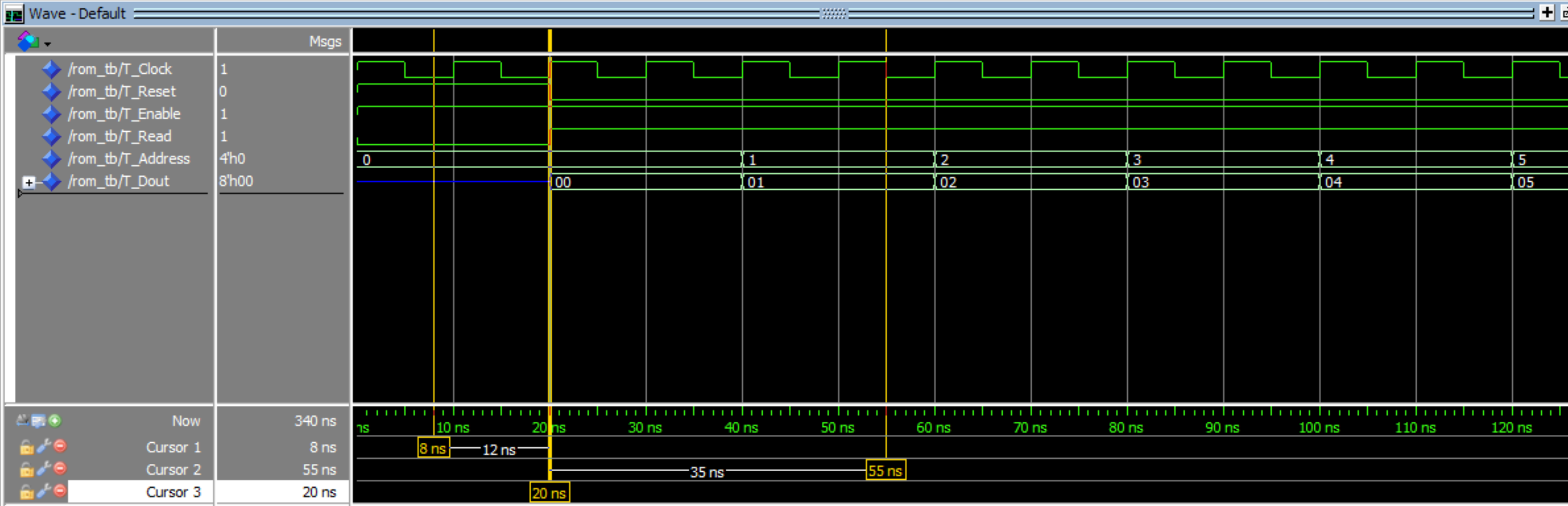
بعد در یک loop مقدار address را هر 20 نانو ثانیه 1 واحد ببیشتر می‌کنیم تا روی خانه های حافظه ROM پیمایش داشته باشیم.

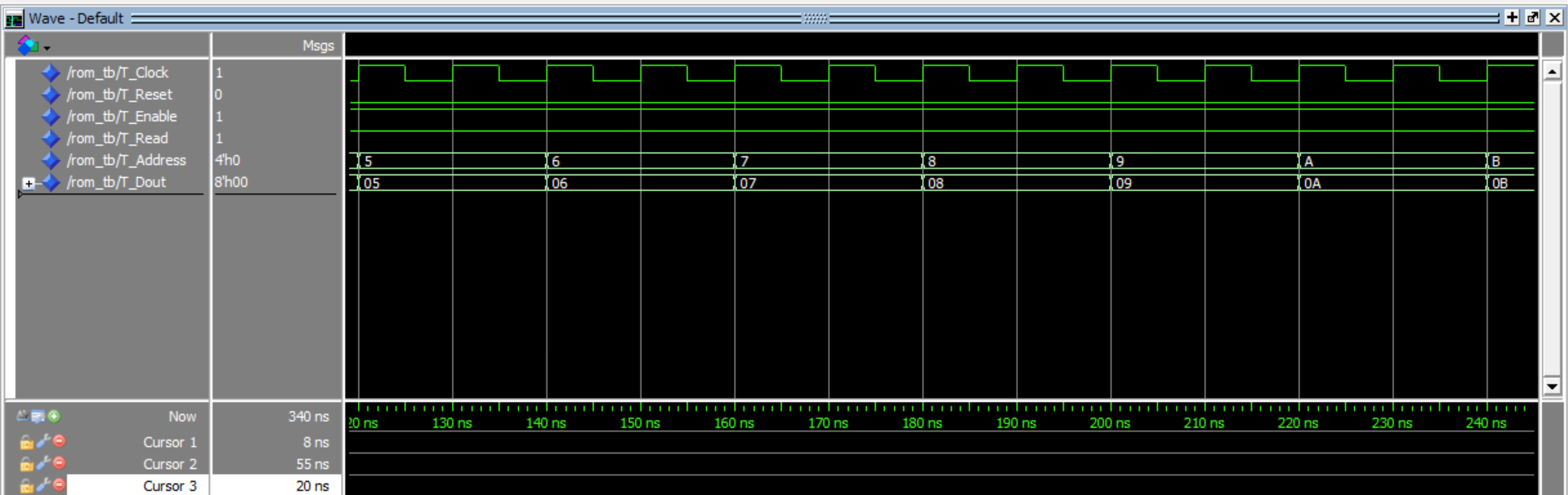
کد تست بنچ به صورت زیر است:

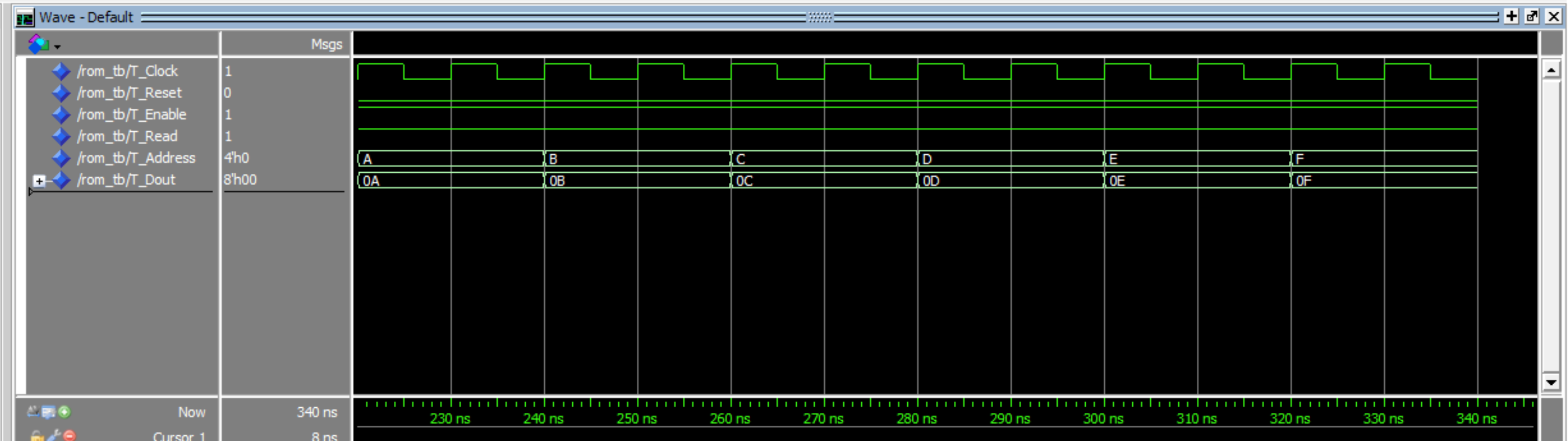
|  |
| --- |
| ------------------------------------------------------------- -- Test Bench for 16\*8 ROM module  -------------------------------------------------------------   library IEEE; use IEEE.std\_logic\_1164.all; use IEEE.std\_logic\_arith.all; use ieee.std\_logic\_unsigned.all;   entity ROM\_TB is end ROM\_TB;    architecture TB of ROM\_TB is  component ROM is port(  clk : in std\_logic;  rst : in std\_logic;  enable : in std\_logic;  read : in std\_logic;  address : in integer range 0 to 15;  dout : out std\_ulogic\_vector(7 downto 0) ); end component;  --init signals signal T\_Clock, T\_Reset, T\_Enable, T\_Read : std\_logic; signal T\_Address : integer range 0 to 15; signal T\_Dout : std\_ulogic\_vector(7 downto 0);  begin --port mapping      U\_ROM: ROM port map (  T\_Clock,   T\_Reset,   T\_Enable,   T\_Read,   T\_Address,   T\_Dout     );      clock\_process: process          --clock     begin         T\_Clock<='1';         wait for 5 ns;         T\_Clock<='0';         wait for 5 ns;     end process;       process      begin  --init reset as 1 so we are not in any specific state  T\_Enable <= '1';  T\_Read <= '0';  T\_Reset <= '1';  T\_Address <= 0;  wait for 20 ns;     --reset = 0 and read = 1 so we can read data from ROM  T\_Reset <= '0';  T\_Read <= '1';    --loop on ROM array addresses  for i in 0 to 31 loop      wait for 20 ns;      T\_Address <= T\_Address + 1;   end loop;   wait;     end process;  end TB; |

## Wave خروجی برنامه

Wave خروجی Test Becnh برنامه ما به صورت زیر است:

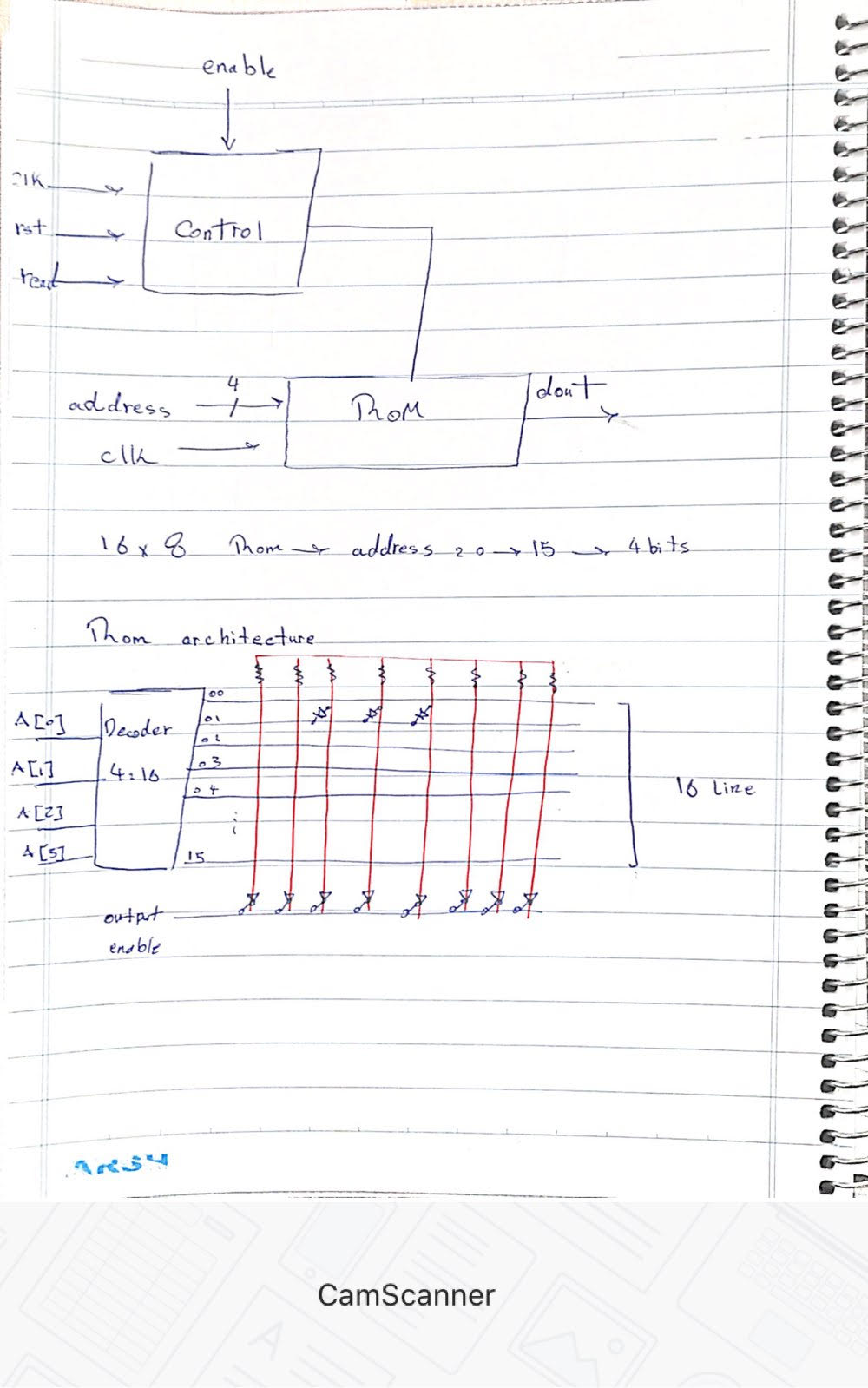






همانطور که در Wave خروجی مشاهده می‌کنید اعداد خانده شده از روی ROM از 0 الی 15 است و برنامه ما خروجی درست نشان داده است.

## شماتیک ROM



همانطور که مشاهده می‌کنید واحد کنترلی اجازه خروج داده را از ROM می‌دهد و خود ROM از یک دیکودر 4 به 16 استفاده می‌کند چون 16x8 هست و ادرس ها 4 بیتی هستند.

ROM:

