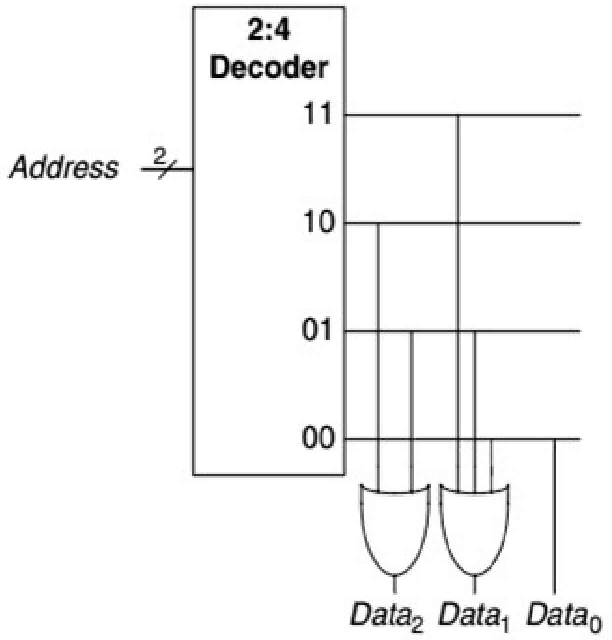
Lesson 17

Memory

Mohd Saufy Rohmad
Technical Trainer and Consultant

Read Only Memory (ROM)

- In theory, ROMs can only be used to read memory.
- A typical ROM is shown in the figure.
- A ROM stores data.
- Each data has a particular address.
- Those addresses can be used to access the datas.
- Datas cannot be written in a ROM



Read Only Memory (ROM)

- We have provided a ROM that can store 16 datas.
- Each data is 8-bit wide.
- The datas are stored in an array called mem_array.
- This array can be accessed with a 4-bit address called RAdrxI.
- Note that, address 0, 1 and 10 to15 have 00000000.
- Thats why they are all declared with others => "00000000" in VHDL.
- We included one register for the read address called, RAdrxR.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity rom is
port (clk: in std_logic ;
      rst: in std_logic;
      RAdrxI: in unsigned(3 downto 0);
      RDataxDO : out std_logic_vector( 7 downto 0 )) ;
end;
architecture rom_arch of rom is
signal RAdrxR : unsigned(3 downto 0);
type mem_array is array ( 0 to 15 ) of std_logic_vector( 7 downto 0 ) ;
```

```
constant mem: mem_array := (
2 => "11111111" ,
3 => "11010101" ,
4 => "01101000" ,
6 => "10011011" ,
8 => "01101101" ,
9 => "00110111" ,
others => "00000000" ) ;
begin
process (clk,rst)
begin
if (rst= '0') then
RAdrxR <= (others=>'0');
elsif ClkxCI'event and ClkxCI = '1' then
RAdrxR <= RAdrxI;
end if;
end process;
RDataxDO <= mem(to_integer(RAdrxR));
end rom_arch;
```

A ROM can be used for instance in instruction memory, where you don’t have to write the data.

Random Access Memory (RAM)

- RAM is slightly more complex than ROM because data can be written on it.
- Thats why there is no need of a pre-stored datas in a RAM code.
- The reading part of the RAM code is simply done by,
- RDataxDO <= mem(to_integer(RAdrxR));
- There is a register for read address similar to the ROM part.
- It is possible to also write in the same mem_array.
- The writing process is done inside a process.
- There is a WExI flag that allows the writing process. Note that, a read enable is not available here.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity ram_array2 is
generic (Row: integer := 16;
Word: integer := 5);
port (
ClkxCI: in std_logic;
RstxCI: in std_logic;
WExI: in std_logic; -- active low
RAdrxI: in unsigned(3 downto 0);
WAdrxI: in unsigned(3 downto 0);
WDataxDI: in STD_LOGIC_VECTOR(Word-1 downto 0);
RDataxDO: out STD_LOGIC_VECTOR(Word-1 downto 0));
end;
```

```
architecture synth2 of ram_array2 is
type mem_array is array (Row-1 downto 0) of
STD_LOGIC_VECTOR(Word-1 downto 0);
signal mem: mem_array;
signal RAdrxR: unsigned(3 downto 0);
begin
process (ClkxCI,RstxCI)
begin
if (RstxCI = '0') then
RAdrxR <= (others=>'0');
elsif ClkxCI'event and ClkxCI = '1' then
RAdrxR <= RAdrxI;
end if;
end process;
```

```
process (ClkxCI,RstxCI,WdataxDI,WadrxI,WexI)
begin
if (RstxCI = '0') then
mem <= (others=>(others=>'0'));
elsif ClkxCI'event and ClkxCI = '1' then
if WExI = '0' then
    mem(to_integer(unsigned(WAdrxI))) <= WDataxDI;
end if;
end if;
end process;
RDataxDO <= mem(to_integer(RAdrxR));
end synth2;
```

Thank You