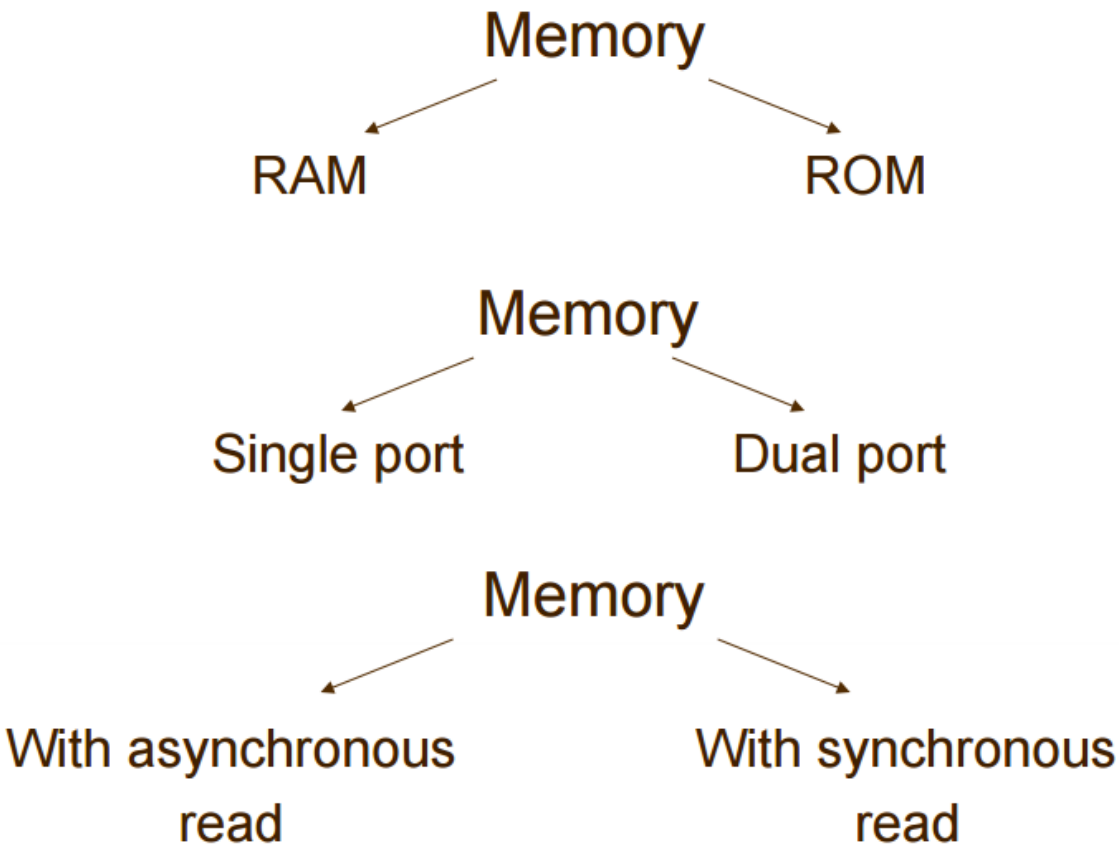
Verilog Notes - 161 Lab

Jose M. Rodriguez
jrodr050@ucr.edu
Chung 464

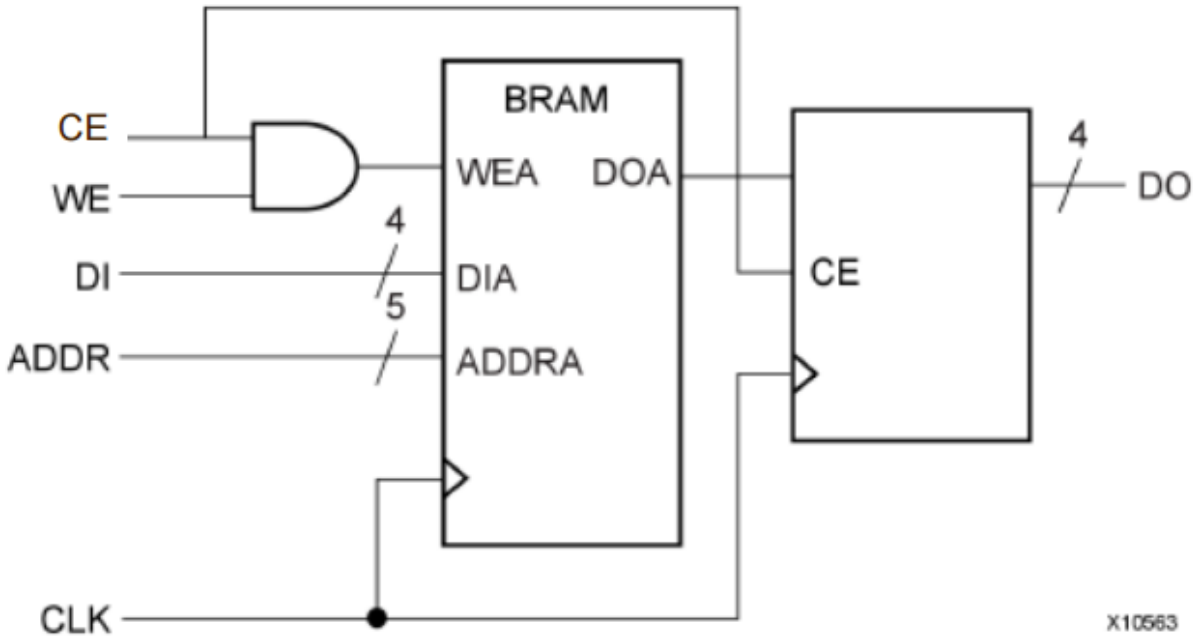
Embedded System Lab
Spring 2017
UC Riverside

May 22, 2017

Memory Types (Source http://ece.gmu.edu)



Single Port RAM - Example



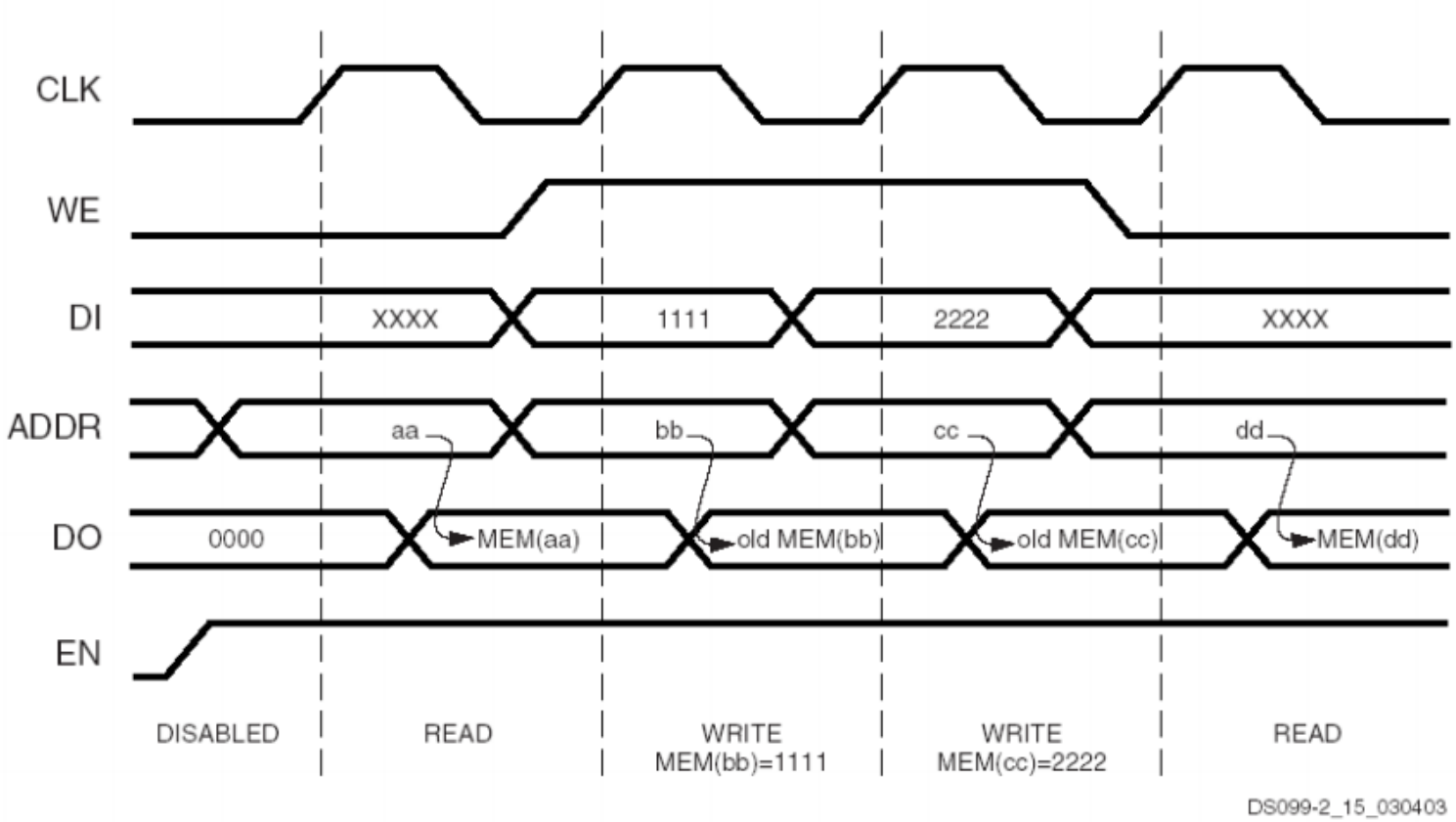
Here **DI** and **DO** are the data in and out. **ADDR** is the read and write address. In addition, **CE** and **WE** are the control and write enable signals.

Single Port RAM (Read First Mode)

```
module v_rams_01 (clk, en, we, addr, di, do);

    input  clk;
    input  we;
    input  en;
    input  [5:0] addr;
    input  [15:0] di;
    output [15:0] do;
    reg    [15:0] RAM [63:0];
    reg    [15:0] do;

    always @(posedge clk)
    begin
        if (en)
        begin
            if (we)
              RAM[addr]<=di;
            do <= RAM[addr];
        end
    end
endmodule
```

Here we declare a memory array with 64 rows each row having 16 bits. **di** and **do** are the data in and data out signals. In addition, **addr** is the read write address.

Block RAM Waveforms – READ_FIRST mode



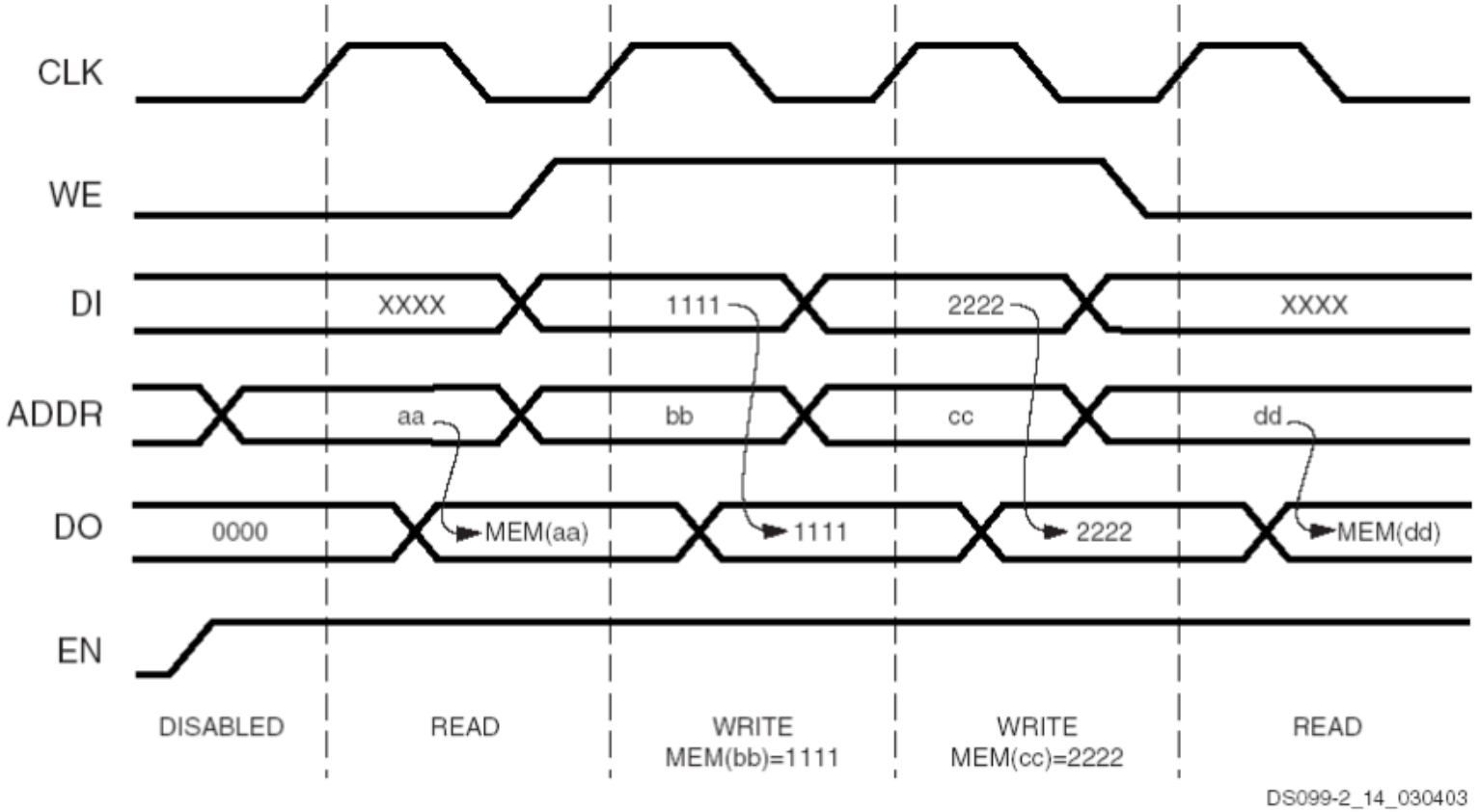
Source (http://ece.gmu.edu)

Single Port RAM (Write First Mode)

```
module v_rams_02a (clk, we, en, addr, di, do);

    input  clk;
    input  we;
    input  en;
    input  [5:0] addr;
    input  [15:0] di;
    output [15:0] do;
    reg    [15:0] RAM [63:0];
    reg    [15:0] do;

    always @(posedge clk)
    begin
        if (en)
        begin
            if (we)
            begin
                RAM[addr] <= di;
                do <= di;
            end
            else
                do <= RAM[addr];
        end
    end
endmodule
```

In the figure, when the signal write enable **we** is high, we write to the memory array. Otherwise we read from address **addr**.

Block RAM Waveforms – WRITE_FIRST mode

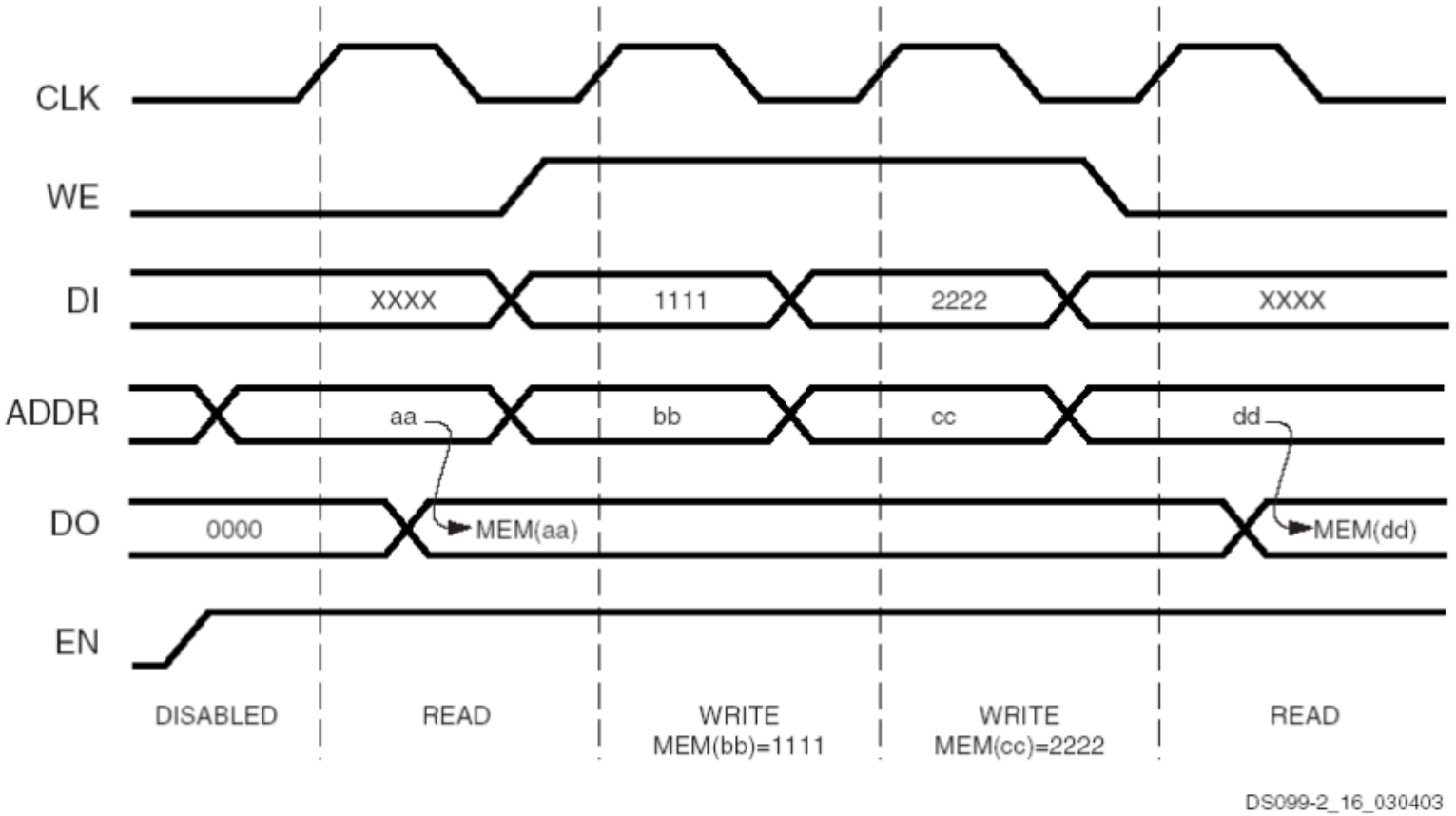


Single Port RAM (No-Change Mode)

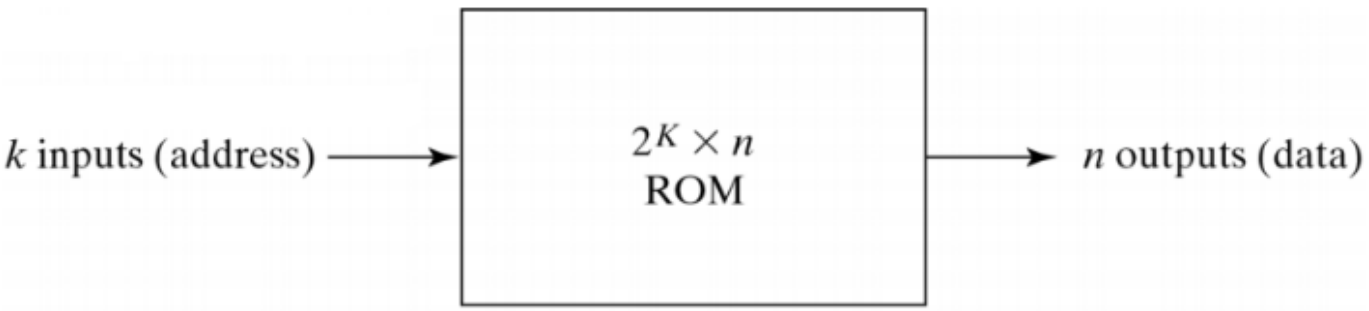
```
module v_rams_03 (clk, we, en, addr, di, do);

    input  clk;
    input  we;
    input  en;
    input  [5:0] addr;
    input  [15:0] di;
    output [15:0] do;
    reg    [15:0] RAM [63:0];
    reg    [15:0] do;

    always @(posedge clk)
    begin
        if (en)
        begin
            if (we)
              RAM[addr] <= di;
            else
              do <= RAM[addr];
        end
    end

endmodule
```

In the figure, when the signal write enable **we** is high, we execute a write. When **we** is low, we execute a read.

Block RAM Waveforms – NO_CHANGE mode



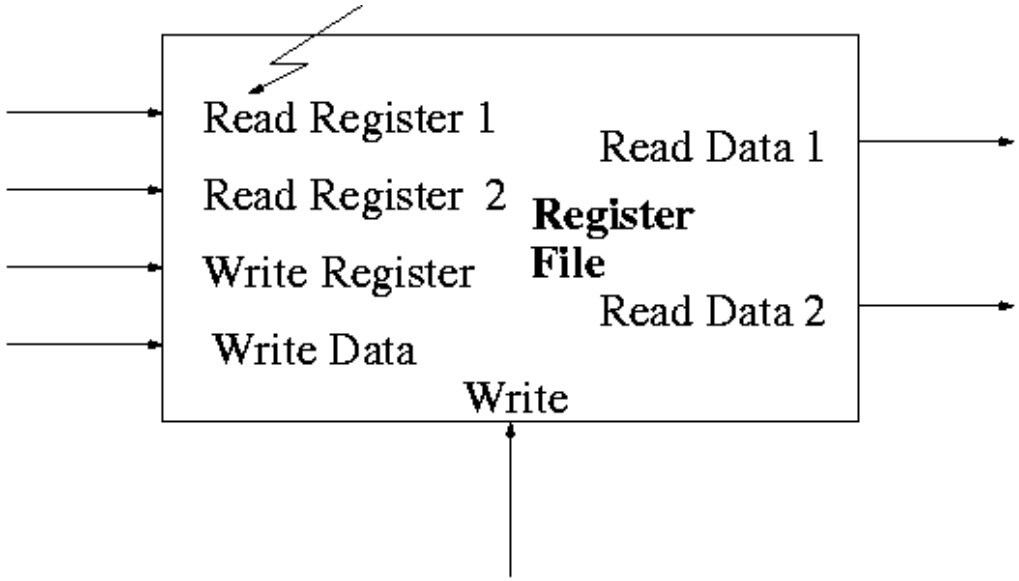
Read Only Memories ROMs



Read Only Memories ROMs

```
 5
 6 module sqrt8rom #(
 7  parameter    NBITS = 32 ,
 8  parameter    Q   = 8
 9 )
10 (
11 input wire clk ,
12 input wire enable ,
13 input wire [Q-1:0 ] rd_addr,
14 output reg  valid ,
15 output reg  [NBITS-1:0 ] rd_dout
16 ) ;
17
18 (* rom_style = "block" *)  reg [Q-1:0] mem [255:0] ;
19
20 // ----------------------------------------
21 //Initializing the ROM contents
22 // ----------------------------------------
23
24 initial begin
25         $readmemb("../verilog/sqrt8.coe", mem, 0, 255);
26 end
27
28 // Read with the clock
29 always @(posedge clk) begin
30         valid   <= enable ;
31         rd_dout <= { {NBITS-Q{1'b0}} , { mem[rd_addr] } };
32
33 end
34
35 endmodule
36
```

Register Files



Register File with 2 read ports and 1 write port

Register Files

```
module regfile(input         clk,
               input         RegWrite,
               input  [4:0]  ra1, ra2, wa,
               input  [31:0] wd,
               output [31:0] rd1, rd2);

  reg [31:0] rf[31:0];

 // three ported register file
 // read two ports combinationally
 // write third port on rising edge of clock
 // register 0 hardwired to 0

  always @(posedge clk)
    if (RegWrite) rf[wa] <= wd;

  assign rd1 = (ra1 != 0) ? rf[ra1] : 0;
  assign rd2 = (ra2 != 0) ? rf[ra2] : 0;

endmodule
```

Review - RAMs and ROMs

In this mini-lecture, we have covered the following key topics.

- Verilog Memory Declaration (reg [W-1:0] datamem [D-1:0])
- Verilog Single Port RAMs (Read First, Write First Mode)
- Verilog Read Only Memories
- Verilog Register Files

Questions ...