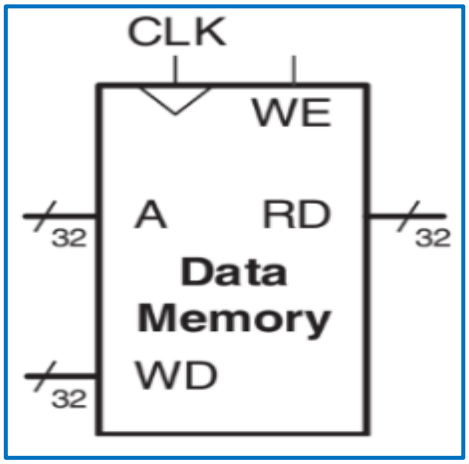
Data Memory Block

1- الهدف هوا اننا نعمل Memory عشان نحط فيها الـ Data اللي مش Frequently acceded من الـ CPU لان دي بنحطها في الـRegister file.

2- الـ Data Memory Block بيكون ليها كذا دخل وخرج واحد :-



Inputs :

1- **we** : Write Enable (we ? write : reed) [1 bit].

2- **clk** : Clock Signal [1 bit].

3- **a** : The Address of a memory location [32 bits].

4- **wd** : Write Data [32 bits].

Output :

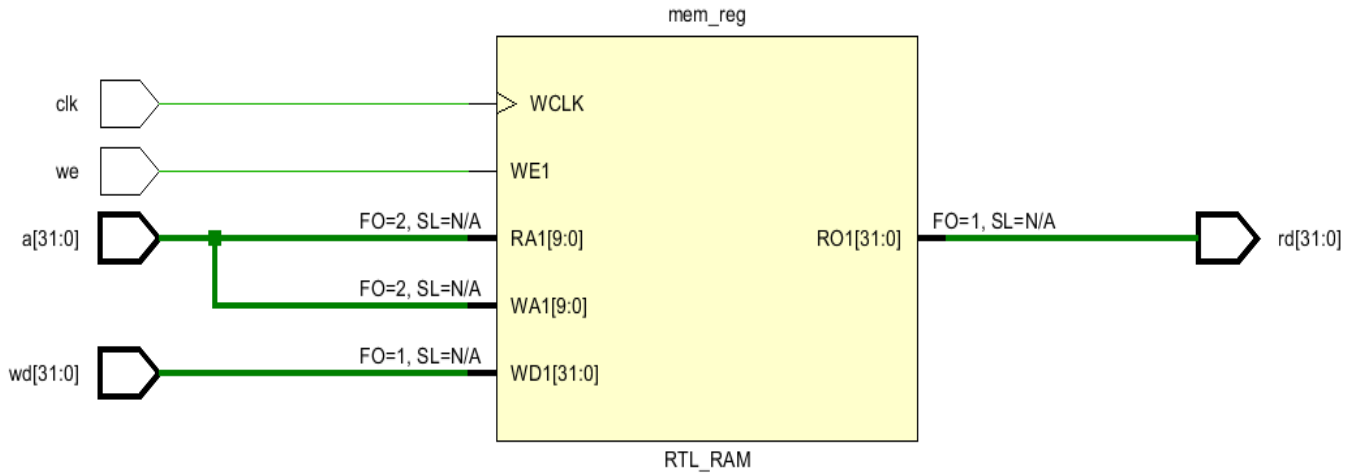
➔ **rd** : read Data [32 bits].

📖 الـ clock بتأثر فقط علي الـ wd ولكنها لا تتدخل في الـ rd بمعني ان الـ wd هي sequential و الـ rd تكون combinational.

3- The RTL Code :-

```
1   `timescale 1ns / 1ps
2
3   module Data_Memory(clk, we, a, wd, rd);
4
5       input wire clk;
6       input wire we;              // Write enable
7       input wire [31:0] a;      // Address input
8       input wire [31:0] wd;     // Write data input
9       output reg [31:0] rd;     // Read data output
10
11      // Memory declaration
12      reg [31:0] RAM [63:0];
13
14      // Continuous assignment for read data
15      always @(*) begin
16          rd = RAM[a[31:2]];  // Word-aligned read
17      end
18
19      // Write to memory
20      always @(posedge clk) begin
21          if (we) begin
22              RAM[a[31:2]] <= wd;
23          end
24      end
```

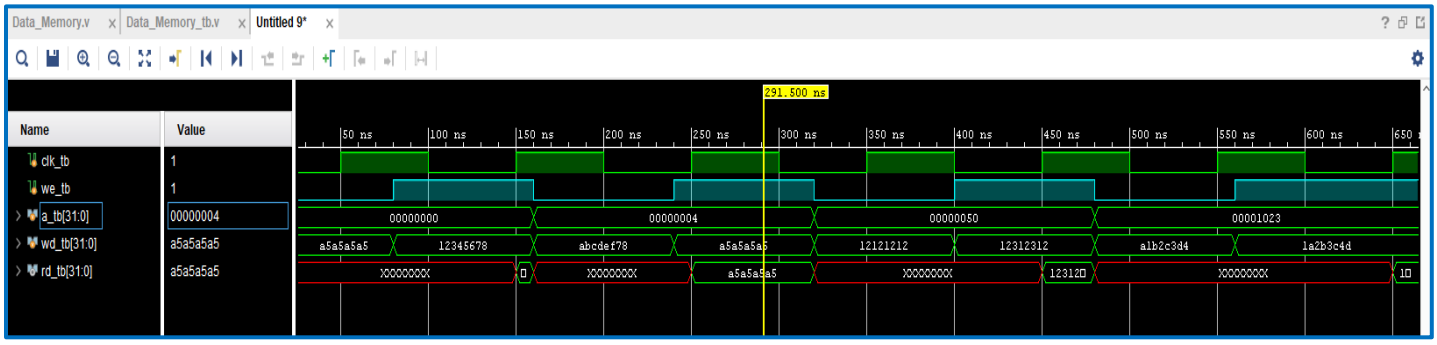
4- Elaborated design:-



5- Testbench :-

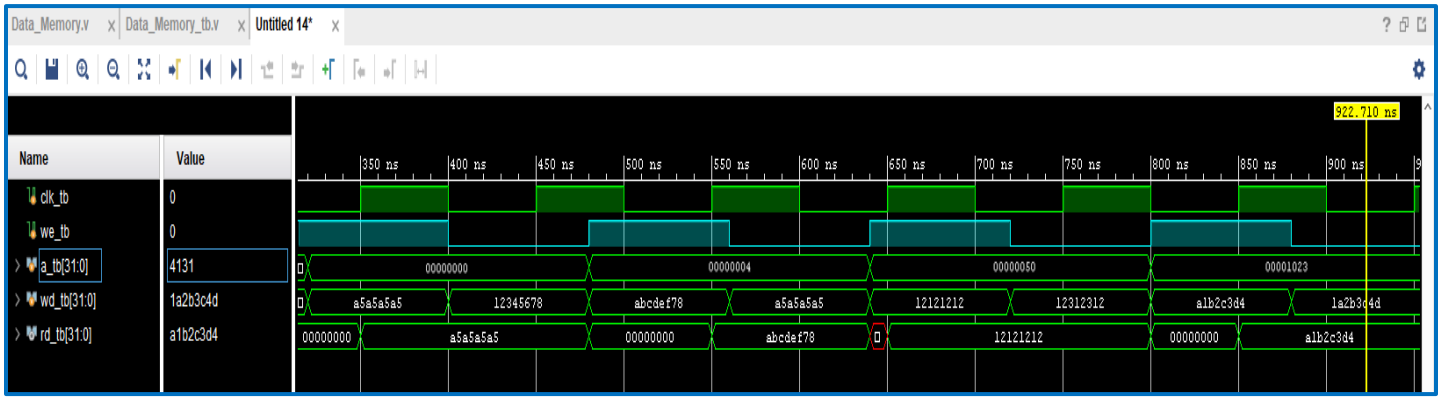
```
`timescale 1ns / 1ps

module Data_Memory_tb();
    // 1) Declare local reg and wire identifiers with _tb suffix
    reg clk_tb = 0;                 // The clock signal for testbench
    reg we_tb;                      // Write Enable signal for testbench
    reg [31:0] a_tb;                // Address  for testbench
    reg [31:0] wd_tb;               // Write Data for testbench
    wire [31:0] rd_tb;              // Read Data for testbench

    // 2) Instantiate the module under test
    Data_Memory uut (
        .clk(clk_tb),
        .we(we_tb),
        .a(a_tb),
        .wd(wd_tb),
        .rd(rd_tb)
    );

    // 3) Generate stimuli, using initial and always
    // Clock generation
    always
    begin
        #50 clk_tb = ~clk_tb;

    end
    //Test Cases
    initial
    begin
        a_tb = 32'h0;      wd_tb = 32'hA5A5A5A5;        we_tb = 0;   #80;
        a_tb = 32'h0;      wd_tb = 32'h12345678;        we_tb = 1;   #80;

        a_tb = 32'h4;      wd_tb = 32'hABCDEF78;        we_tb = 0;   #80;
        a_tb = 32'h4;      wd_tb = 32'hA5A5A5A5;        we_tb = 1;   #80;

        a_tb = 32'h50;     wd_tb = 32'h12121212;        we_tb = 0;   #80;
        a_tb = 32'h50;     wd_tb = 32'h12312312;        we_tb = 1;   #80;

        a_tb = 32'h1023;   wd_tb = 32'hA1B2C3D4;        we_tb = 0;   #80;
        a_tb = 32'h1023;   wd_tb = 32'h1A2B3C4D;        we_tb = 1;   #80;

    end
endmodule
```

←نتيجة الـ **Simulation** :



-تظهر قيم الـ rd_tb علي انها unknown بسبب ان هذة الـ Addresses في الـ Data memory لا تحتوي علي قيم مسبقة مخزنه بها , ولكن بعد عملية الـ Write بها ظهرت مباشرة لانه اصبح هناك قيمة مخزنة في هذا الـ Adress.

←**وللتاكيد تم تخزين قيمة مسبقة = 0 في هذة الـ Addresses**:



Made by: Hossam Ahmed Seyam