CPE 233: RAT assignment 3
Prof. Bridget Benson
Luis Gomez, Jared Rocha

BBD

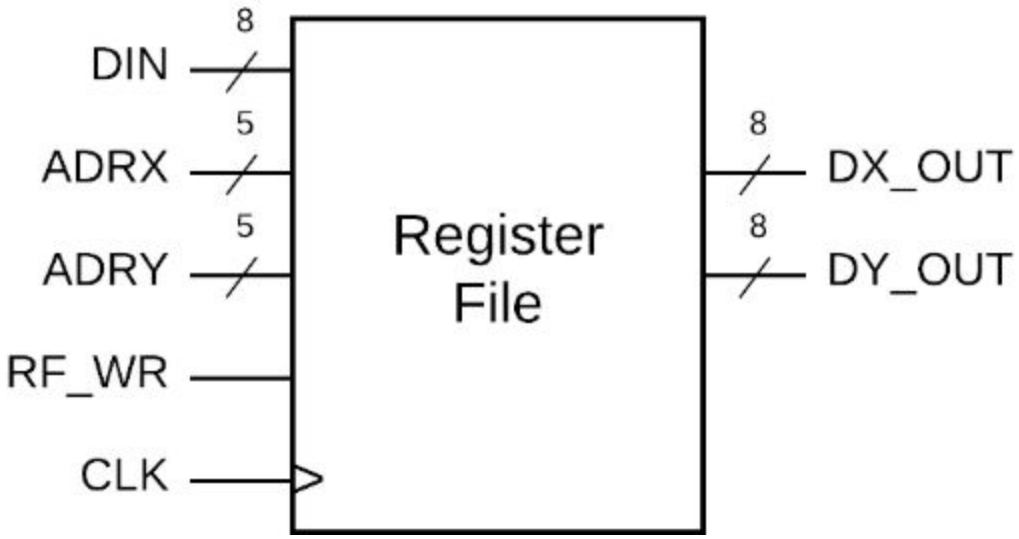

Figure 1: Black Box Diagram Register File

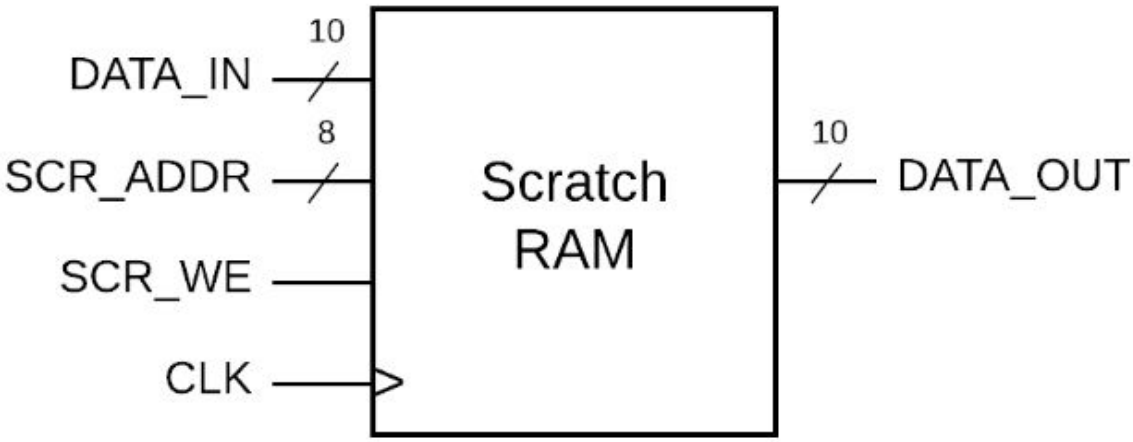

Figure 2: Black Box Diagram Scratch Ram

Behavior

In this assignment, we built two RAM memory modules in system verilog a register file and a scratch ram. The following will explain each.

Register File:

A simple RAM memory module that holds 32 registers that can each store 8 - bits. Dual port RAM allowing reading from two locations or addresses simultaneously. Address inputs ADRX and ADRY with a value between 0-31, and output values saved in registers to DX_OUT and DY_OUT. Only a signal register can be saved per clock cycle with ADRX being the only save address. DATA_IN is a 8-bit value saved to the location of ADRX. Data is read asynchronously, while data is saved synchronously with the rising clock edge and only if RF_WR input is 1.

Scratch Ram:
Is a RAM memory module that has 256 locations that can store 10 bits. Used for stack memory and temporary saving data. Data can be transferred from Scratch Ram to Register file. Single port RAM so only one address SCR_ADDR can be read from and written into. DATA_IN is read asynchronously and written synchronously on the rising clock edge and input SCR_WE is 1.

Structural Design

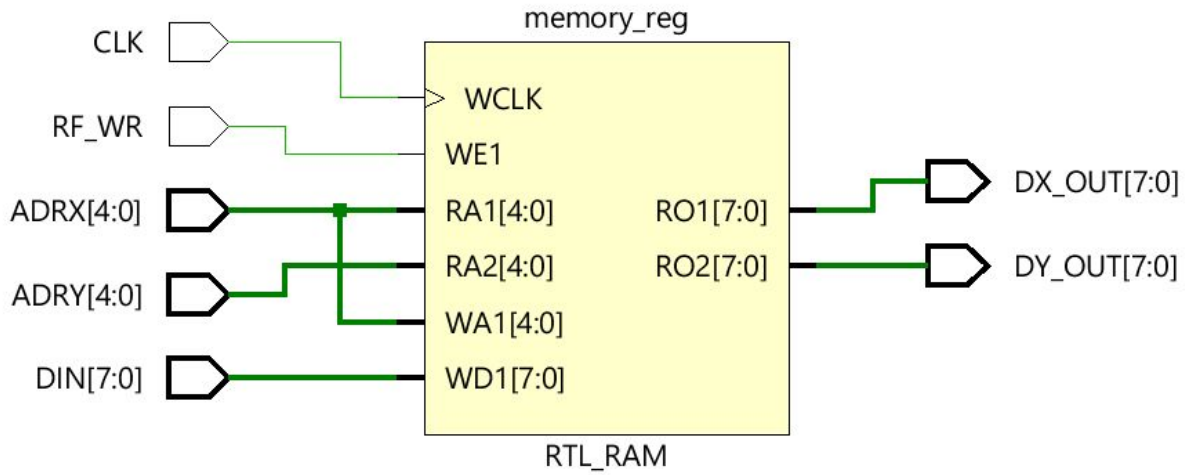

Figure 3: Register File Schematic Drawing

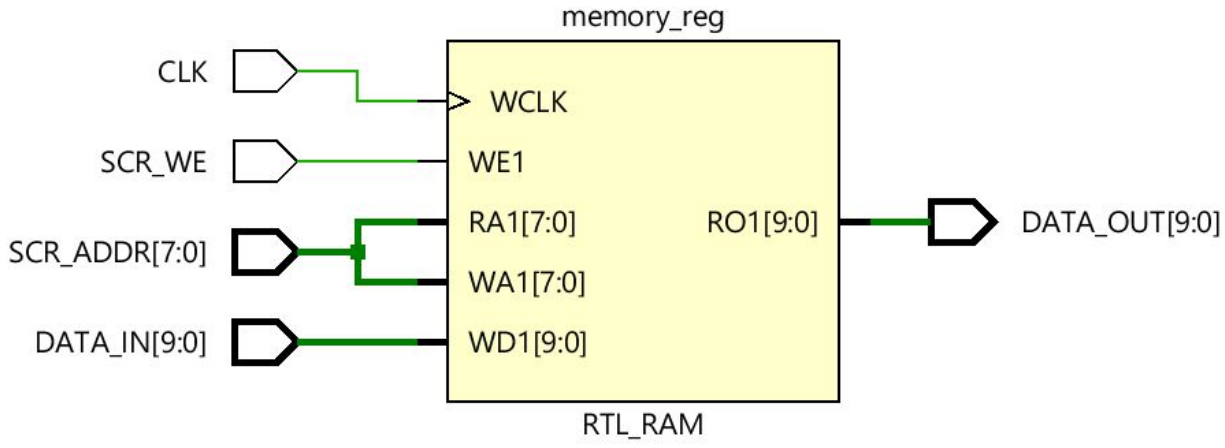

Figure 4: Scratch Ram Schematic Drawing

Verification

Register File: We did two test. One with RF_WR = 1 and one with RF_WR = 0.

Test 1: With RF_WR = 1 to write to memory. DIN = i, ADRX = i. With clock cycle of 100 nanoseconds. We ran a for loop to write to all locations of memory and read all locations. Testing ADRX to write to memory and have DX_OUT read ADRX value.

Test 2: With RF_WR = 0 to just read from memory but, not write. DIN = i, ADRY = i. With a clock cycle of 100 nanoseconds. We ran a for loop to read all locations of memory. Testing to ensure memory was not written too.

Test Table:

Test #	RF_WR	DIN	ADRX	ADRY	Explanation
1	1	i	i	N/A	Testing writing and reading to memory
2	0	i	N/A	i	Testing reading but not writing to memory

Figure 5: Register File Test Table

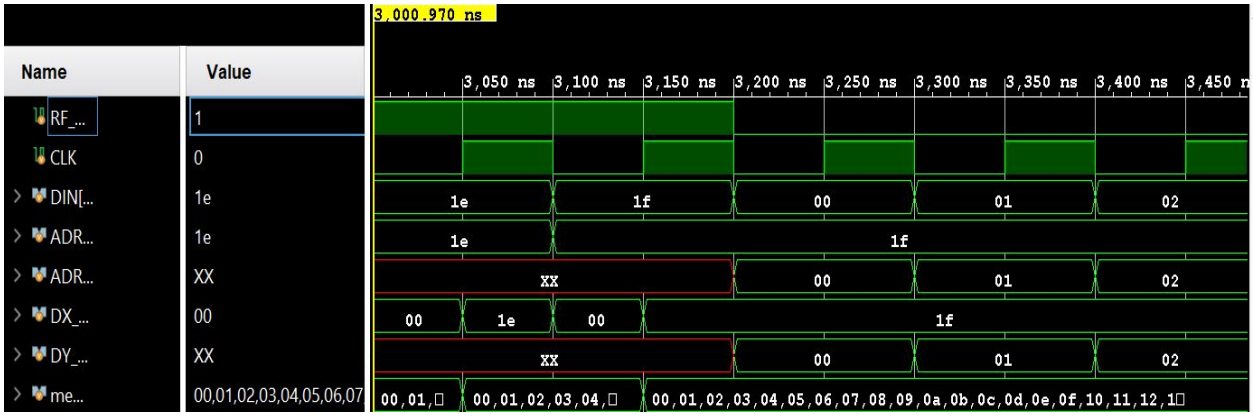

Figure 6: Register File Simulation

Register File Test Bench Code:

```
module register_file_tb();
       logic RF_WR, CLK;
       logic [7 : 0] DIN;
       logic [4 : 0] ADRX;
       logic [4 : 0] ADRY;
       logic [7 : 0] DX_OUT;
       logic [7 : 0] DY_OUT;
       register_file DUT(.*);

       always
       begin
           CLK = 0; #50;            // 50 nanoseconds
           CLK = 1; #50;
       end
           initial begin
           int i;
           RF_WR = 1;               // allows for memory to written to
           for (i =0; i<32; i++)    // loop to go through all of memory
           begin
              DIN = i;              // all memory is written to by i position
              ADRX = i;
              #100;                 // 100 nanoseconds clock cycle
           end
           RF_WR = 0;               // memory will not be written to
           for (i =0; i<32; i++)
           begin
              DIN = i;              // read memory for position i
              ADRY = i;
              #100;
endmodule
```

Figure 7: Register File Test Bench Code

Scratch Ram: We did two test. One for writing and one for reading.

Test 1: We tested for writing to all memory locations and, reading of output.

Test 2: We test for reading of output without writing to memory locations.

Test Table:

Test #	SCR_WE	DATA_IN	SCR_ADDR	Explanation
1	1	i	i	Testing writing and reading to memory
2	0	i	i	Testing reading but not writing to memory

Figure 8: Scratch Ram Test Table

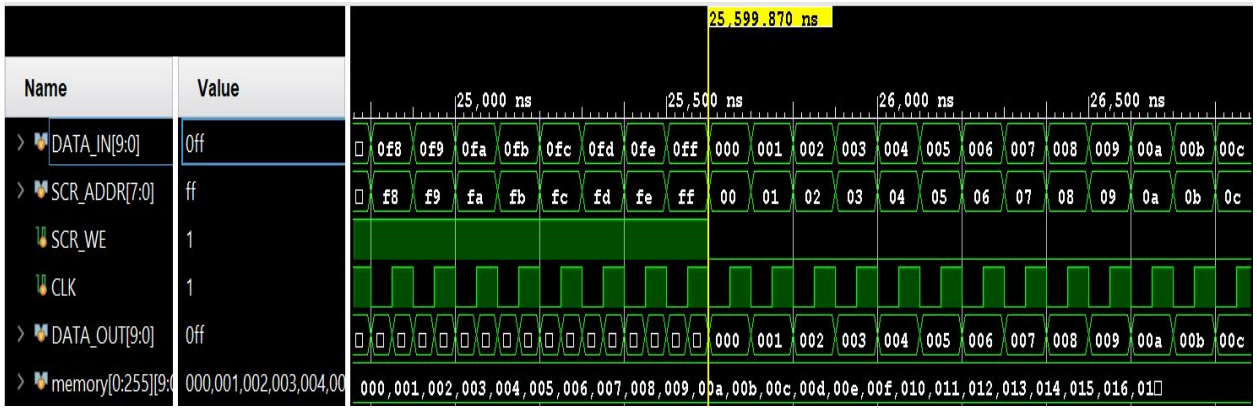

Figure 9: Scratch Ram Simulation

Scratch Ram Test Bench Code:

```
module scratch_ram_tb();
       logic [9:0] DATA_IN;
       logic [7:0] SCR_ADDR;
       logic SCR_WE, CLK;
       logic [9:0] DATA_OUT;
       scratch_ram DUT(.*);

       always
       begin
           CLK = 0; #50;
           CLK = 1; #50;
       end
           initial begin
           int i;
           SCR_WE = 1;               // allows for memory to written to
           for (i =0; i<256 ; i++)  // loop to go through all memory
           begin
              DATA_IN = i;
              SCR_ADDR = i;
              #100;
           end
           SCR_WE = 0;               // memory can not be written to
           for (i =0; i<256 ; i++)
           begin
              DATA_IN = i;
              SCR_ADDR = i;
              #100;
            end
            end
endmodule
```

Figure 10: Scratch File Test Bench Code

Source Code

Register File

```
module register_file(
input RF_WR, CLK,
input [7 : 0] DIN,
input [4 : 0] ADRX,
input [4 : 0] ADRY,
output [7 : 0] DX_OUT,
output [7 : 0] DY_OUT
);

logic [7 : 0] memory [0 : 31];      // array for 32 8-bit registers
initial begin
     int i;
     for (i =0; i<32; i++) begin    // loop to cycle through array
          memory[i] = 0;            // start location zero in array
     end
end

always_ff @ (posedge CLK)
begin
   if (RF_WR==1)                    // allows for memory to be written to
       memory[ADRX] = DIN;          // data input written to register of value
End                                 // ADRX
assign DX_OUT = memory[ADRX];       // output register address ADRX to DX_OUT
assign DY_OUT = memory[ADRY];       // output register address ADRY to DY_OUT
Endmodule
```

Figure 11: Register File Source Code

Scratch Ram

```
module scratch_ram(
input [9:0] DATA_IN,
input [7:0] SCR_ADDR,
input SCR_WE, CLK,
output [9:0] DATA_OUT
);

logic [9:0] memory [0:255];          // array for 256 10-bit memory holders
initial begin
     int i;
     for (i =0; i<256; i++) begin    // loop through array
          memory[i] = 0;             // start at 0 in array
     end
end

always_ff @ (posedge CLK)
begin
   if (SCR_WE==1)                    // allows for memory to written to
       memory[SCR_ADDR] = DATA_IN;   // input data to memory location SCR_ADDR
end
assign DATA_OUT = memory[SCR_ADDR]; // output data
Endmodule
```

Figure 12: Scratch Ram Source Code