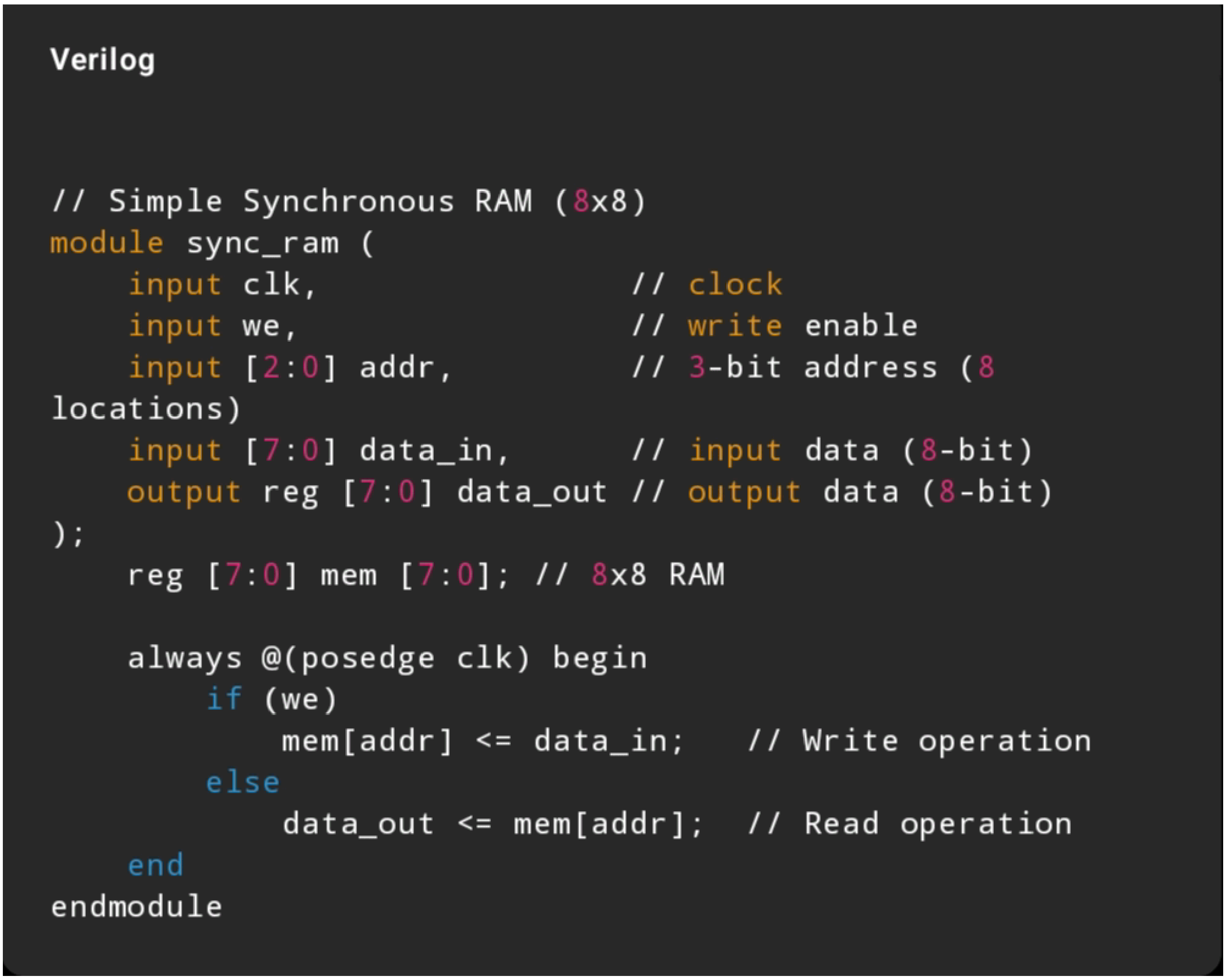
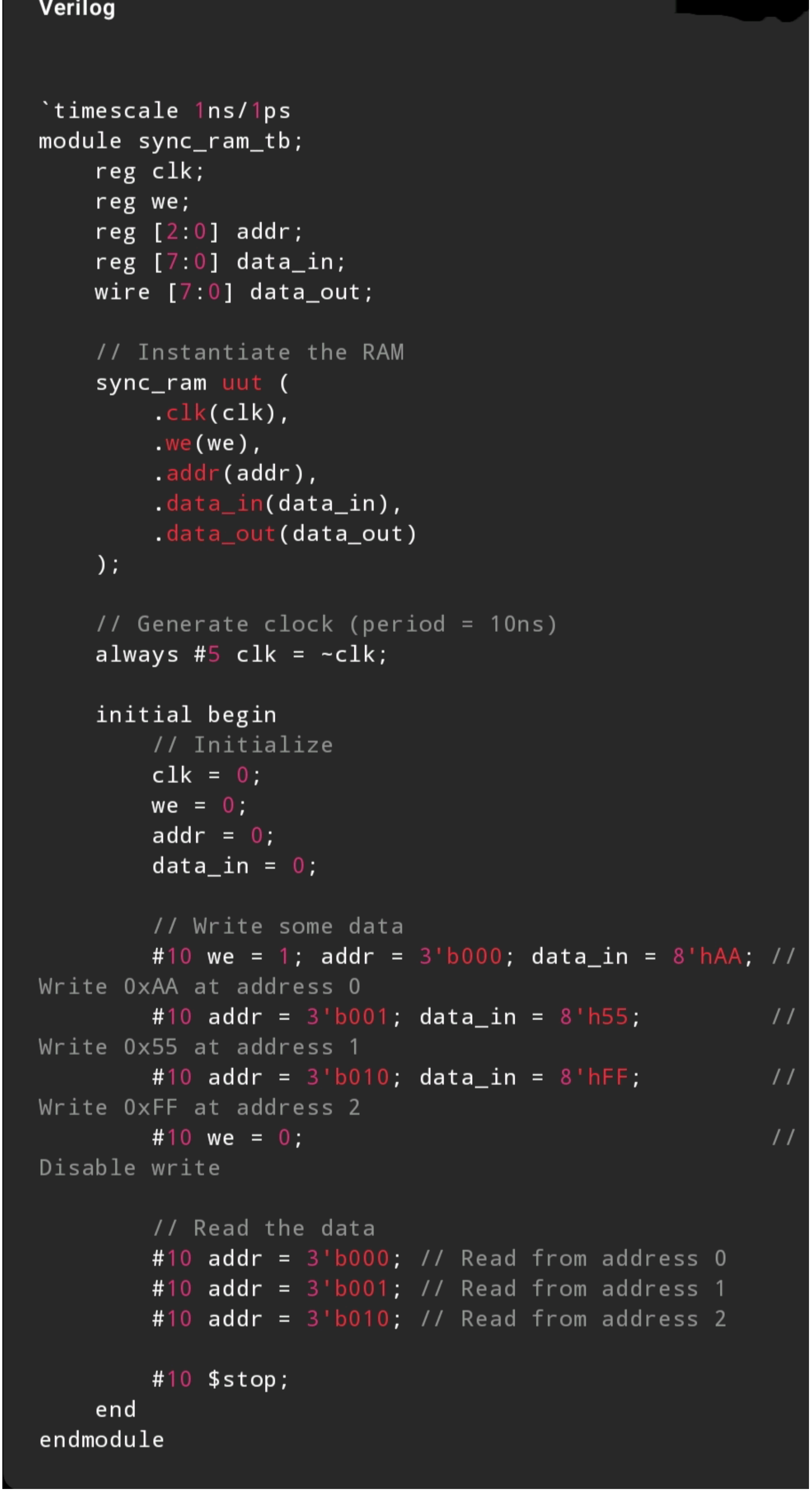
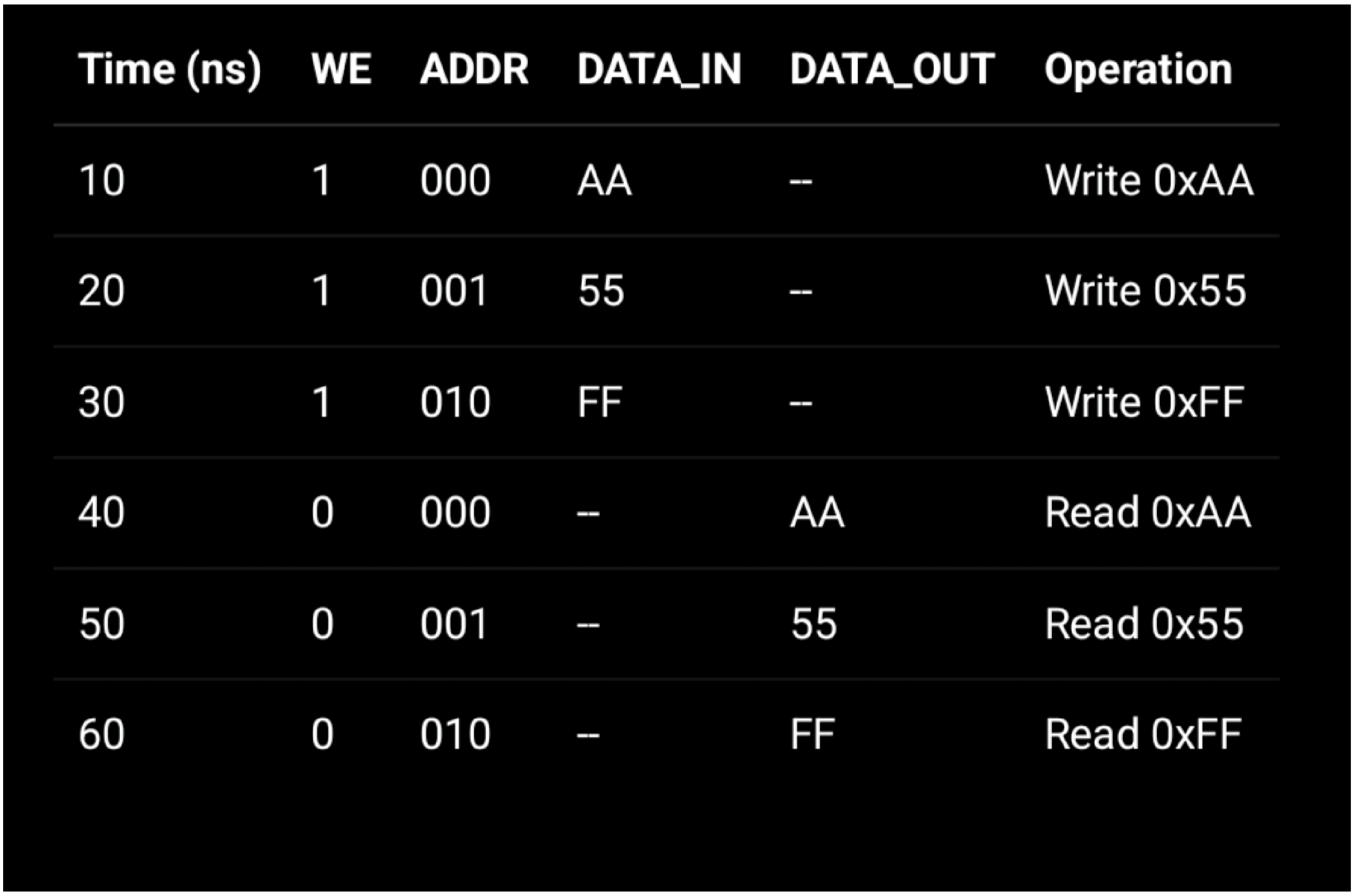

Verilog

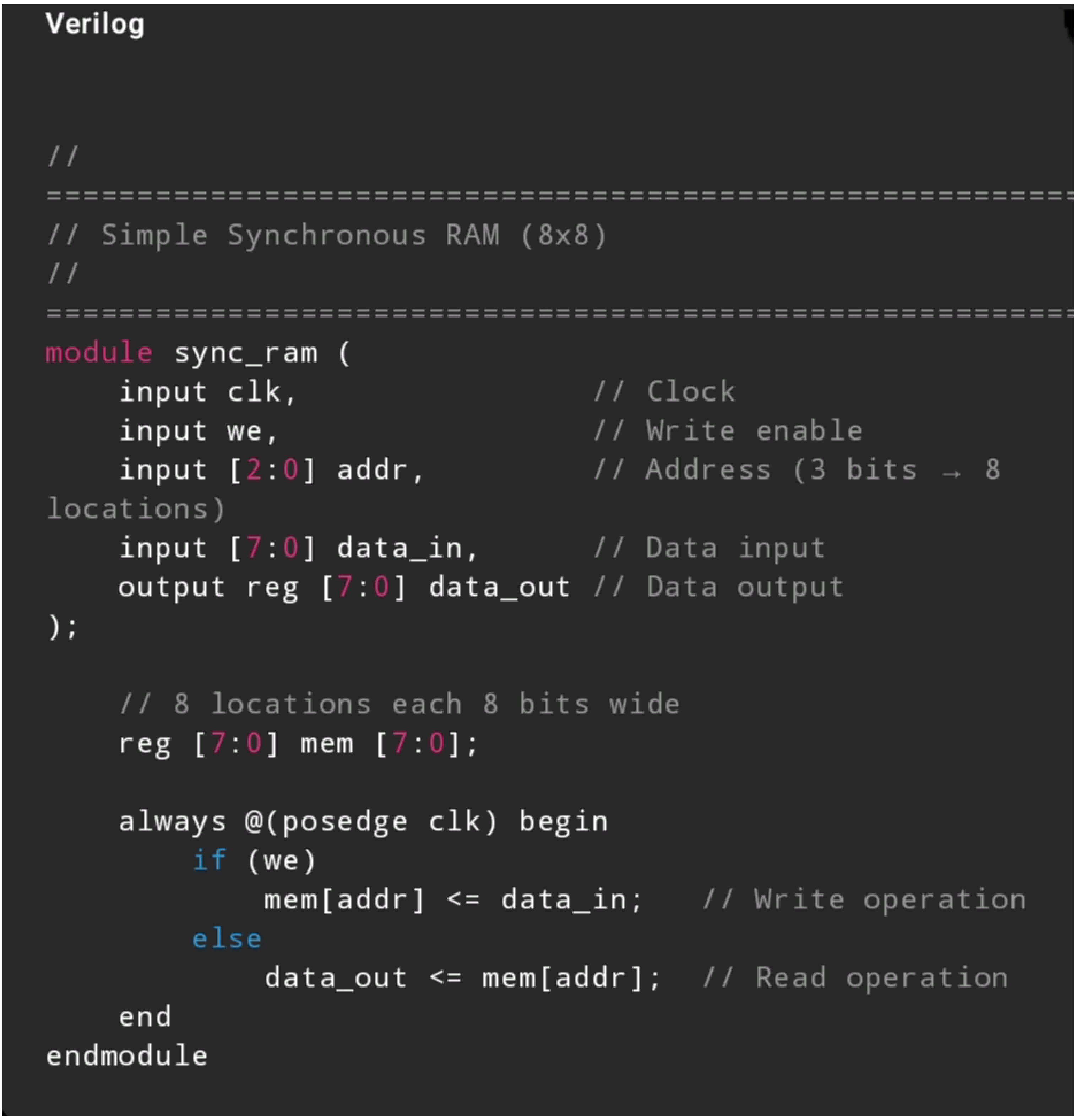
```
// Simple Synchronous RAM (8x8)
module sync_ram (
    input clk,                 // clock
    input we,                  // write enable
    input [2:0] addr,          // 3-bit address (8
locations)
    input [7:0] data_in,       // input data (8-bit)
    output reg [7:0] data_out // output data (8-bit)
);
    reg [7:0] mem [7:0]; // 8x8 RAM

    always @(posedge clk) begin
        if (we)
            mem[addr] <= data_in;   // Write operation
        else
            data_out <= mem[addr];  // Read operation
    end
endmodule
```


Verilog

```verilog
`timescale 1ns/1ps
module sync_ram_tb;
    reg clk;
    reg we;
    reg [2:0] addr;
    reg [7:0] data_in;
    wire [7:0] data_out;

    // Instantiate the RAM
    sync_ram uut (
        .clk(clk),
        .we(we),
        .addr(addr),
        .data_in(data_in),
        .data_out(data_out)
    );

    // Generate clock (period = 10ns)
    always #5 clk = ~clk;

    initial begin
        // Initialize
        clk = 0;
        we = 0;
        addr = 0;
        data_in = 0;

        // Write some data
        #10 we = 1; addr = 3'b000; data_in = 8'hAA; //
Write 0xAA at address 0
        #10 addr = 3'b001; data_in = 8'h55;         //
Write 0x55 at address 1
        #10 addr = 3'b010; data_in = 8'hFF;         //
Write 0xFF at address 2
        #10 we = 0;                                 //
Disable write

        // Read the data
        #10 addr = 3'b000; // Read from address 0
        #10 addr = 3'b001; // Read from address 1
        #10 addr = 3'b010; // Read from address 2

        #10 $stop;
    end
endmodule
```



Time (ns)	WE	ADDR	DATA_IN	DATA_OUT	Operation
10	1	000	AA	--	Write 0xAA
20	1	001	55	--	Write 0x55
30	1	010	FF	--	Write 0xFF
40	0	000	--	AA	Read 0xAA
50	0	001	--	55	Read 0x55
60	0	010	--	FF	Read 0xFF



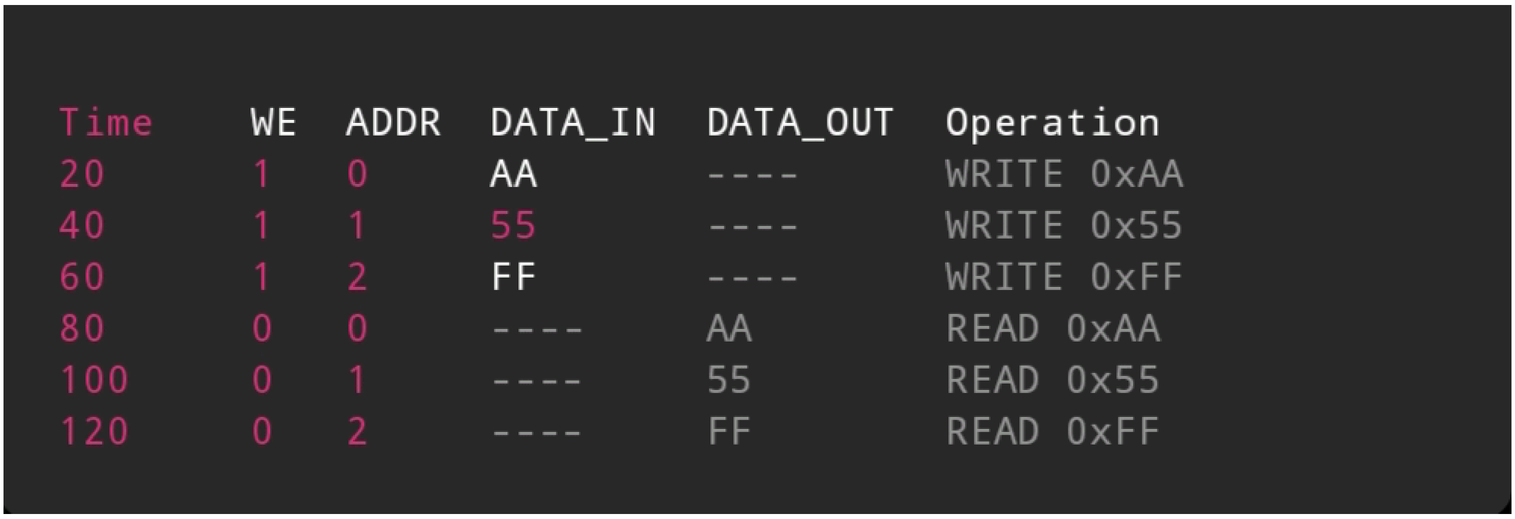
Verilog

```verilog
//
==========================================================
// Simple Synchronous RAM (8x8)
//
==========================================================
module sync_ram (
    input clk,                  // Clock
    input we,                   // Write enable
    input [2:0] addr,           // Address (3 bits → 8
locations)
    input [7:0] data_in,        // Data input
    output reg [7:0] data_out // Data output
);

    // 8 locations each 8 bits wide
    reg [7:0] mem [7:0];

    always @(posedge clk) begin
        if (we)
            mem[addr] <= data_in;    // Write operation
        else
            data_out <= mem[addr];  // Read operation
    end
endmodule
```

Verilog

```verilog
`timescale 1ns/1ps
module sync_ram_tb;

    reg clk;
    reg we;
    reg [2:0] addr;
    reg [7:0] data_in;
    wire [7:0] data_out;

    // Instantiate the RAM
    sync_ram uut (
        .clk(clk),
        .we(we),
        .addr(addr),
        .data_in(data_in),
        .data_out(data_out)
    );

    // Clock generation: 10ns period
    always #5 clk = ~clk;

    initial begin
        // Initialize
        clk = 0;
        we = 0;
        addr = 0;
        data_in = 0;


$display("Time\tWE\tADDR\tDATA_IN\tDATA_OUT\tOperation");

        // Write operations
        #10 we = 1; addr = 3'b000; data_in = 8'hAA; //
Write 0xAA
        #10 $display("%0t\t%b\t%0d\t%h\t----\tWRITE 0xAA",
$time,we,addr,data_in);

        #10 addr = 3'b001; data_in = 8'h55; // Write 0x55
        #10 $display("%0t\t%b\t%0d\t%h\t----\tWRITE 0x55",
$time,we,addr,data_in);

        #10 addr = 3'b010; data_in = 8'hFF; // Write 0xFF
        #10 $display("%0t\t%b\t%0d\t%h\t----\tWRITE 0xFF",
$time,we,addr,data_in);

        // Read operations
        #10 we = 0;
        #10 addr = 3'b000;
        #10 $display("%0t\t%b\t%0d\t----\t%h\tREAD 0xAA",
$time,we,addr,data_out);
```

```
        #10 addr = 3'b001;
        #10 $display("%0t\t%b\t%0d\t----\t%h\tREAD 0x55",
$time,we,addr,data_out);

        #10 addr = 3'b010;
        #10 $display("%0t\t%b\t%0d\t----\t%h\tREAD 0xFF",
$time,we,addr,data_out);

        #20 $stop;
    end
endmodule
```



```
Time       WE   ADDR   DATA_IN   DATA_OUT   Operation
20         1    0      AA        ----       WRITE 0xAA
40         1    1      55        ----       WRITE 0x55
60         1    2      FF        ----       WRITE 0xFF
80         0    0      ----      AA         READ 0xAA
100        0    1      ----      55         READ 0x55
120        0    2      ----      FF         READ 0xFF
```