**Synchronous Static RAM**

1. **Introduction**

**Memory** is a basic element in any system whether the memory is **volatile** or **non-volatile**. In this example, a **volatile memory unit** is designed in the form of a **Synchronous Static RAM**. **Static Random-Access Memory (SRAM)** is a type of **semiconductor memory** that uses **bi-stable latching circuitry** to store each bit. The term **Static** differentiates it from **Dynamic RAM (DRAM)** which must be periodically refreshed. **SRAM** retains data, but it is still **volatile** as data is lost when the power to the memory unit is cut off.

1. **Verilog Module**

Figure 1 presents the Verilog module of the **Synchronous SRAM**. This **Synchronous SRAM** can store **eight 8-bit** values. The **Synchronous SRAM** module consists of a **8-bit data input** line, **dataIn** and a **8-bit data output** line, **dataOut**. The module uses an **8-bit address** line, **Addr** to locate the position of data-byte within the memory array. With an **8-bit address** line a **256-unit deep SRAM** can be addressed, but in this example, an **8-unit deep SRAM** is designed for simplicity. The module is clocked using the **1-bit input clock** line **Clk**. The module also has a **1-bit chip select** line, **CS**.  
  
The **1-bit RD** line is used to signal a data read operation on the **Synchronous SRAM** and the **1-bit WE** line is used to signal a data write operation on the **Synchronous SRAM**. Both the **RD** and **WE** lines are **active high**.

|  |
| --- |
| [C:\Users\san\Desktop\Notes\Verilog for Beginners_ Synchronous Static RAM_files\Verilog Module.PNG](https://1.bp.blogspot.com/-OCjXQmnCXFk/VDz4MfUVrBI/AAAAAAAAAb0/kz6ilW_omdE/s1600/Verilog%2BModule.PNG) |
| Figure 1. Verilog module of **Synchronous SRAM** |

1. **Verilog Code for Synchronous SRAM *(syncRAM.v)***

|  |
| --- |
| * 1. module syncRAM( dataIn,   2. dataOut,   3. Addr,   4. CS,   5. WE,   6. RD,   7. Clk   8. );   10. // parameters for the width   11. parameter ADR = 8;   12. parameter DAT = 8;   13. parameter DPTH = 8;   14. //ports   15. input [DAT-1:0] dataIn;   16. output reg [DAT-1:0] dataOut;   17. input [ADR-1:0] Addr;   18. input CS,   19. WE,   20. RD,   21. Clk;   23. //internal variables   24. reg [DAT-1:0] SRAM [DPTH-1:0];   25. always @ (posedge Clk)   26. begin   27. if (CS == 1'b1) begin   28. if (WE == 1'b1 && RD == 1'b0) begin   29. SRAM [Addr] = dataIn;   30. end   31. else if (RD == 1'b1 && WE == 1'b0) begin   32. dataOut = SRAM [Addr];   33. end   34. else;   35. end   36. else;   37. end   38. endmodule |
| Figure 2. Verilog Code for **Synchronous SRAM** |

1. **Verilog Test Bench for Synchronous SRAM *(syncRAM\_tb.v)***

|  |
| --- |
| * 1. `timescale 1ns / 1ps   2. module syncRAM\_tb;   3. // Inputs   4. reg [7:0] dataIn;   5. reg [7:0] Addr;   6. reg CS;   7. reg WE;   8. reg RD;   9. reg Clk;   10. // Outputs   11. wire [7:0] dataOut;   12. // Instantiate the Unit Under Test (UUT)   13. syncRAM uut (   14. .dataIn(dataIn),   15. .dataOut(dataOut),   16. .Addr(Addr),   17. .CS(CS),   18. .WE(WE),   19. .RD(RD),   20. .Clk(Clk)   21. );   22. initial begin   23. // Initialize Inputs   24. dataIn = 8'h0;   25. Addr = 8'h0;   26. CS = 1'b0;   27. WE = 1'b0;   28. RD = 1'b0;   29. Clk = 1'b0;   30. // Wait 100 ns for global reset to finish   31. #100;   33. // Add stimulus here   34. dataIn = 8'h0;   35. Addr = 8'h0;   36. CS = 1'b1;   37. WE = 1'b1;   38. RD = 1'b0;   39. #20;   40. dataIn = 8'h0;   41. Addr = 8'h0;   42. #20;   43. dataIn = 8'h1;   44. Addr = 8'h1;   45. #20;   46. dataIn = 8'h10;   47. Addr = 8'h2;   48. #20;   49. dataIn = 8'h6;   50. Addr = 8'h3;   51. #20;   52. dataIn = 8'h12;   53. Addr = 8'h4;   54. #40;   55. Addr = 8'h0;   56. WE = 1'b0;   57. RD = 1'b1;   58. #20;   59. Addr = 8'h1;   60. #20;   61. Addr = 8'h2;   62. #20;   63. Addr = 8'h3;   64. #20;   65. Addr = 8'h4;   66. end   68. always #10 Clk = ~Clk;   70. endmodule |