DAY 11
MEMORY ELEMENTS - I

October 24, 2023

Memory Modeling

In Verilog, memory elements can be modeled using arrays of **reg** data types. These memory elements can represent various types of storage devices, such as registers, RAMs, ROMs, and FIFOs. The most common memory element modeled in Verilog is Random Access Memory (RAM). In this impelmentation we are using a ROM to create a 2-bit compartor. ROM modules also can be used in different FSM implmentations.

First, we derive the Truth Table for a 2-bit comparator. Next, we define these values

$A1A0$	$B1B0$	$A > B$ $A = B$ $A < B$
00	00	010
00	01	001
00	10	001
00	11	001
01	00	100
01	01	010
01	10	001
01	11	001
10	00	100
10	01	100
10	10	010
10	11	001
11	00	100
11	01	100
11	10	100
11	11	010

Table 1: Truth Table

(output) in a seperate text file, which then can be read by verilog using either **readmemb** or **readmemh**. In this case I have used the **readmemb** command as these values are stored in binary format. Alternatively, you can use ".mif" file format to store data for each memory address.

ROM Module

```
module rom16_4(output reg [3:0] data_out,input [3:0] data_addr);

reg [3:0] mem [15:0]; // Memory to hold 4 bit wide 16 memory addresses

initial begin
  $readmemb("mem.txt",mem);
end

always@(*) begin
  data_out = mem[data_addr];
end


endmodule
```

2-bit Comparator Module

```
`include "rom16_4.v"
module comp(output A_eq_B,A_ls_B,A_gr_B,input [1:0] A,B);

// Define the address
wire [3:0] addr;
// Define the data_out
wire [3:0] out;

assign addr = {A,B};

// Instantiate the ROM
rom16_4 ROM(.data_addr(addr),.data_out(out));

// Assign the outputs
assign A_eq_B = out[2];
assign A_ls_B = out[1];
assign A_gr_B = out[3];

endmodule
```

Simulation Results

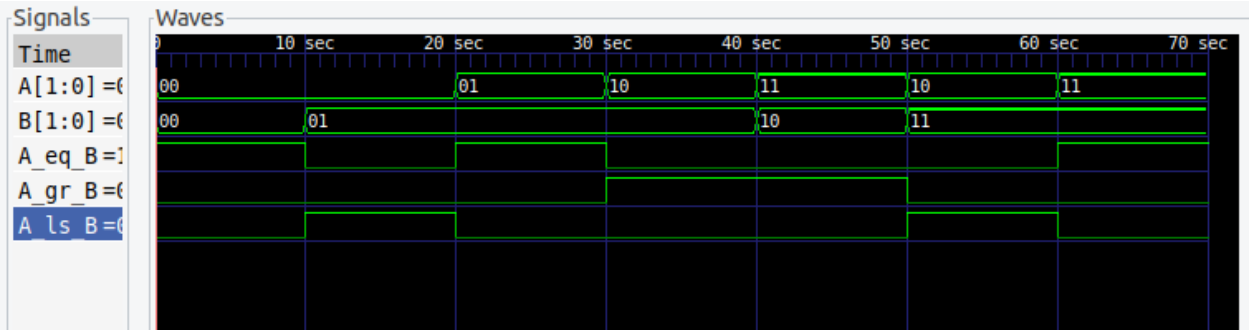

Figure 1: Simulation