Practice Problems (1)

RAM chips are available in a variety of sizes. If the memory unit needed for an application is larger than the capacity of one chip, it is necessary to combine a number of chips in an array to form the required memory size. The capacity of the memory depends on two parameters: the number of words and the number of bits per word. An increase in the number of words requires that we increase the address length. Every bit added to the length of the address doubles the number of words in memory. The increase in the number of bits per word requires that we increase the length of the data input and output lines, but the address length remains the same.

Problem 1.
Suppose that we have many 1Kx8 RAM chips available with us but we need a memory of size 2Kx8 for our application. This can be constructed with two 1Kx8 RAM chips as shown in the Figure below. The 8 input data lines go to both the chips. The outputs must be ORed together to form the common 8 output data lines. (The OR gates are not shown in the diagram.) The 2K word memory requires an 11-bit address. The 10 least significant bits of the address are applied to the address inputs of both the chips.

The most significant bit is applied to a 1x2 decoder. The two outputs of the decoder are applied to the CS inputs of each chip. The memory is disabled when the memory-enable input of the decoder is equal to 0. This causes both the outputs of the decoder to be in the 0 state and none of the chips are selected. When the decoder is enabled, address bit 11 determines the particular chip that is selected. If bit 11 is equal to 0, the first RAM chip is selected. The remaining ten address bits select a word within the chip in the range from 0 to 1023. The next 1024 words are selected from the second RAM chip with an 11-bit address that starts with 1 and follows by the ten bits from the common address lines. The address range for each chip is listed in decimal over its block diagram in the Figure.

In this problem, write a top level module to combine two 1Kx8 RAM chips (given below) to form a 2Kx8 memory.

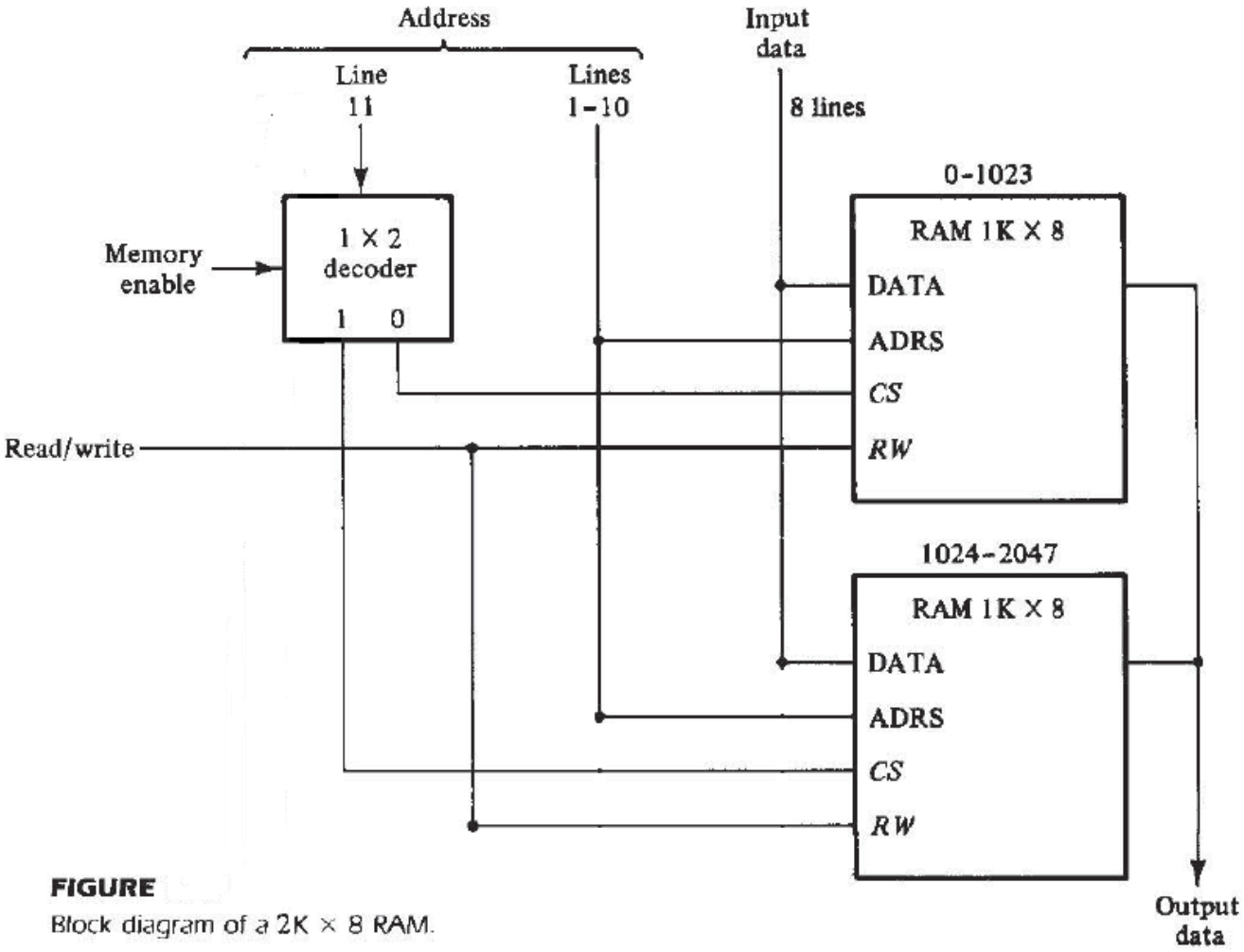

FIGURE
Block diagram of a 2K × 8 RAM.

```
module RAM1 (addr, CS, RW, idata, odata);

      input CS, RW;
      input [9:0] addr;
      input [7:0] idata;
      output [7:0] odata;
      reg [7:0] d_out;
      reg [7:0] Mem1 [0:1023];

      assign odata = (CS && RW)?d_out:8'b0;

      always @(addr or idata or CS or RW)
            if (CS && !RW)
                  Mem1 [addr] = idata;
      always @(addr or CS or RW)
            if (CS && RW)
                  d_out = Mem1 [addr];

      initial
            $readmemh ("memory1.dat", Mem1);

endmodule
```

```
module RAM2 (addr, CS, RW, idata, odata);

      input CS, RW;
      input [9:0] addr;
      input [7:0] idata;
      output [7:0] odata;
      reg [7:0] d_out;
      reg [7:0] Mem2 [0:1023];

      assign odata = (CS && RW)?d_out:8'b0;

      always @(addr or idata or CS or RW)
            if (CS && !RW)
                  Mem2 [addr] = idata;
      always @(addr or CS or RW)
            if (CS && RW)
                  d_out = Mem2 [addr];

      initial
            $readmemh ("memory2.dat", Mem2);

endmodule
```

Problem 2.

As discussed in the class, it is also possible to combine two chips to form a composite memory containing the same number of words but with twice as many bits in each word. Figure below shows the interconnection of two 1Kx8 RAM chips to form a 1Kx16 memory. The 16 input and output data lines are split between the two chips. Both receive the same 10-bit address and the common CS and RW control inputs.

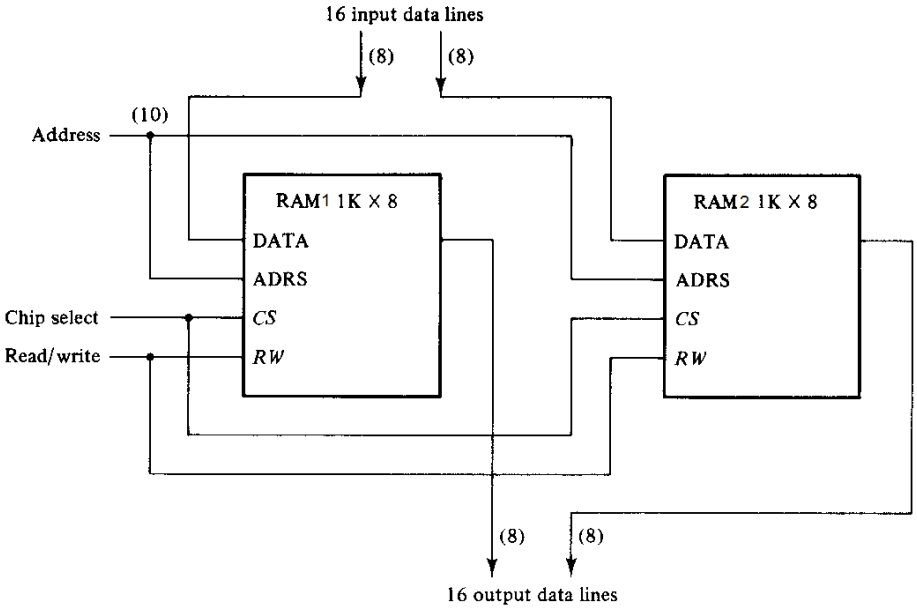

FIGURE
Block diagram of a 1K x 16 RAM.

In this problem, write a top level module to combine two 1Kx8 RAM chips (given below) to form a 1Kx16 memory.

```
module RAM1 (addr, CS, RW, idata, odata);

      input CS, RW;
      input [9:0] addr;
      input [7:0] idata;
      output [7:0] odata;
      reg [7:0] d_out;
      reg [7:0] Mem1 [0:1023];

      assign odata = (CS && RW)?d_out:8'b0;

      always @(addr or idata or CS or RW)
            if (CS && !RW)
                  Mem1 [addr] = idata;
      always @(addr or CS or RW)
            if (CS && RW)
                  d_out = Mem1 [addr];

      initial
            $readmemh ("memory1.dat", Mem1);

endmodule
```

```
module RAM2 (addr, CS, RW, idata, odata);

      input CS, RW;
      input [9:0] addr;
      input [7:0] idata;
      output [7:0] odata;
      reg [7:0] d_out;
      reg [7:0] Mem2 [0:1023];

      assign odata = (CS && RW)?d_out:8'b0;

      always @(addr or idata or CS or RW)
            if (CS && !RW)
                  Mem2 [addr] = idata;
      always @(addr or CS or RW)
            if (CS && RW)
                  d_out = Mem2 [addr];

      initial
            $readmemh ("memory2.dat", Mem2);

endmodule
```