

Memory Computer Science 240

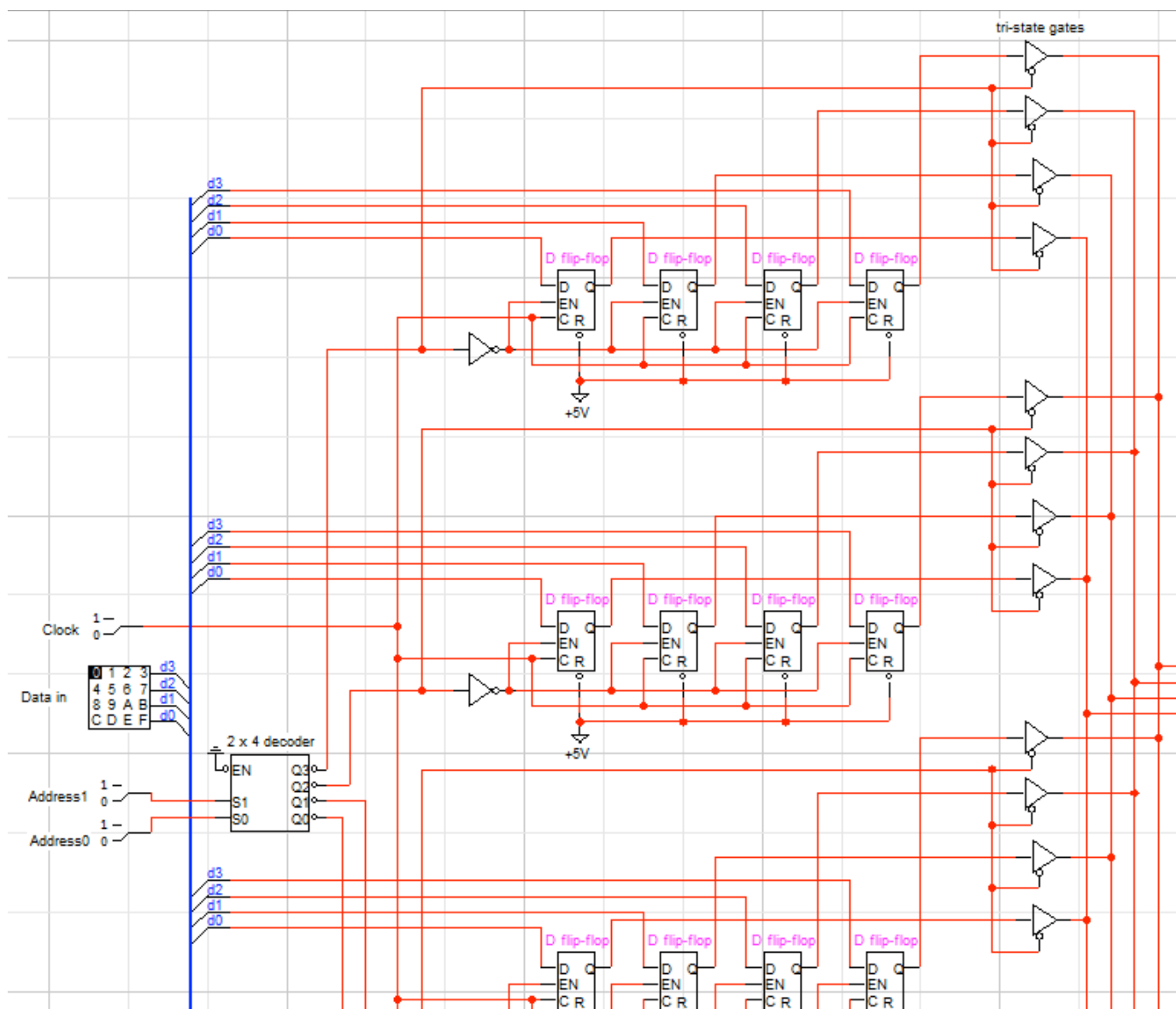
Laboratory 8

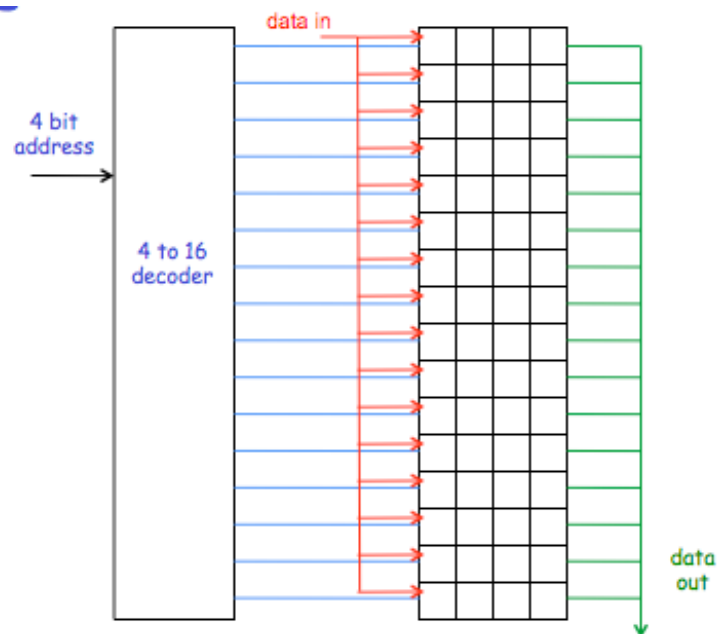
In lab today, you will design and/or investigate the behavior of several memory circuits.

Memory Organization

Exercise 1: On the lab assignment for today, you explained the operation of a smaller version (4x4) of the following 16x4 SRAM memory organization from lecture.

On the version from the assignment, the clock and the tri-state buffers were explicitly shown. As mentioned in lecture, the tri-state buffers are often built in to the latch or flip-flop, and activated through the use of an enable line on the device).





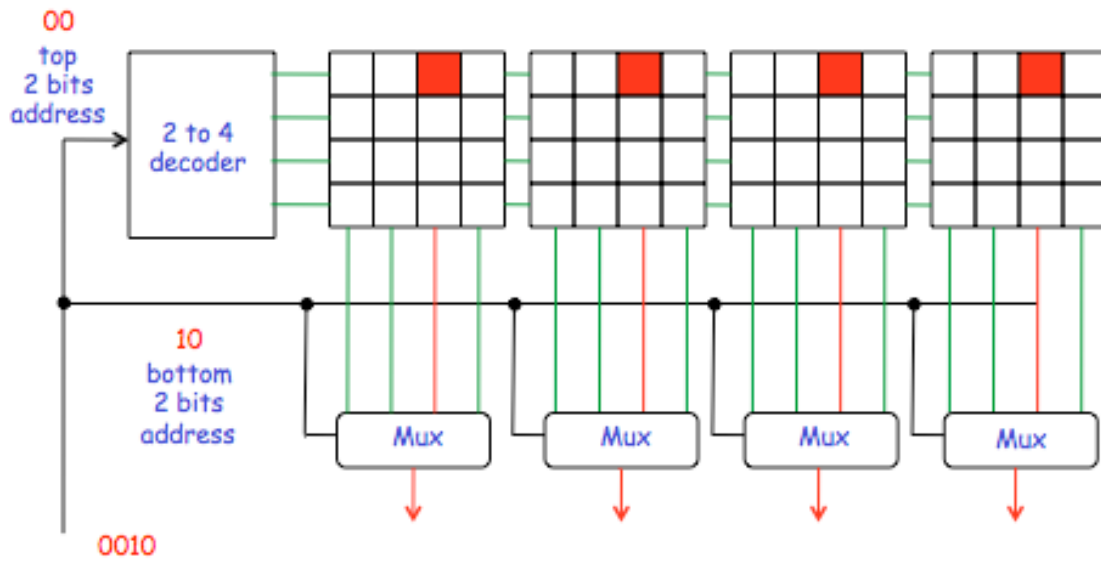
Open the circuit *sram4x4.cct* from the Google group and operate the circuit to store the following values:

Address	Data In (4 bit hex value)
00	A
01	6
10	D
11	2

Set the address and data values, and toggle the clock to 1 and back to 0 for each set of values. Demonstrate to the instructor that you understand how to operate the circuit.

This organization works well for small memories, but does not scale well to larger memories, because too many inputs and outputs are needed as the memory gets larger.

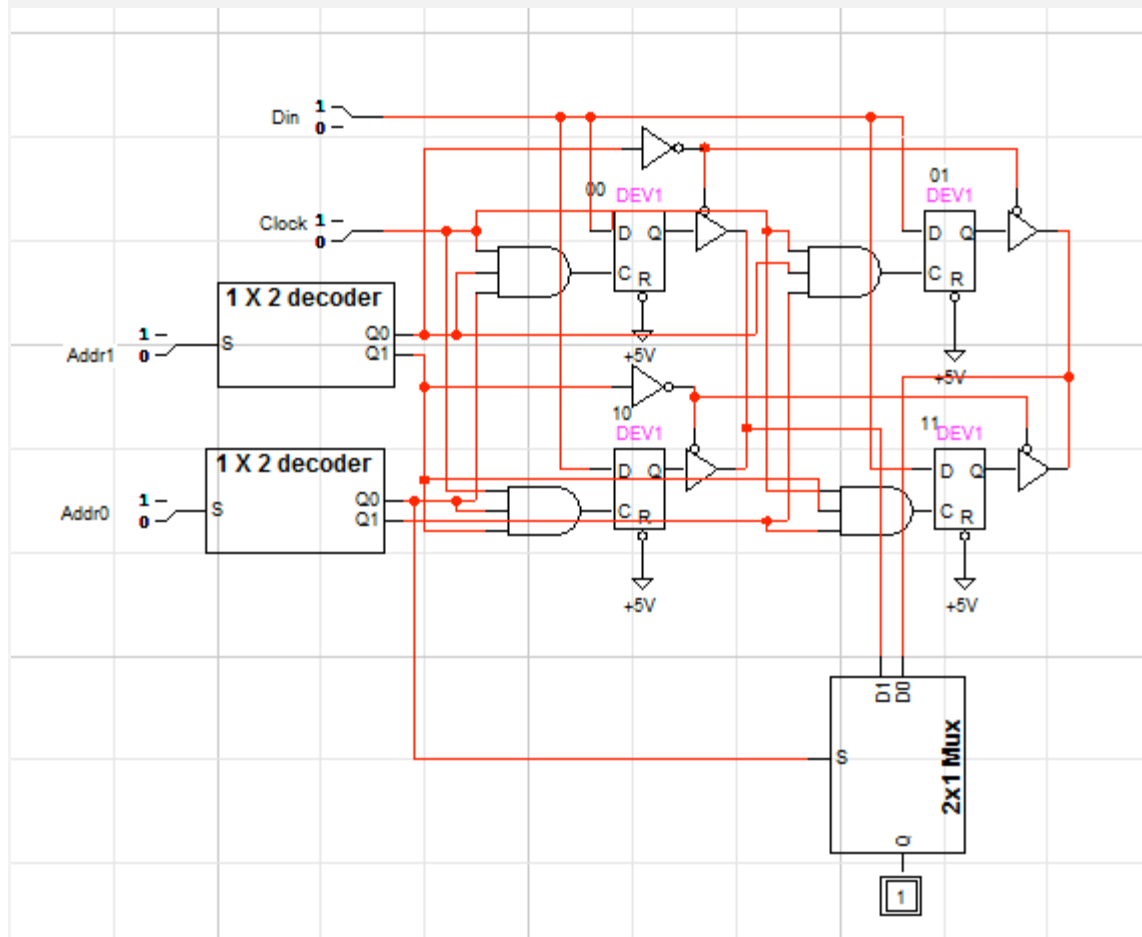
Exercise 2: An alternative memory organization provides some savings for larger memories, as illustrated by the following diagram from lecture:



Implement a 2x2 memory block like the 4x4 shown in the diagram above in LogicWorks.

Start with the circuit *2x2.cct* from the Google group.

Demonstrate to the instructor and paste your circuit here.



Memory Chips

Exercise 3: An actual memory device is the 74189, a memory chip with tri-state outputs built in to the circuit. Investigate the 74189 device by studying its data sheet (available online).

1. How many addresses may be accessed on the 74189?

4 address bits, so $2^4 = 16$ addresses

2. How many bits of data are stored at each address?

4 data bits at each address

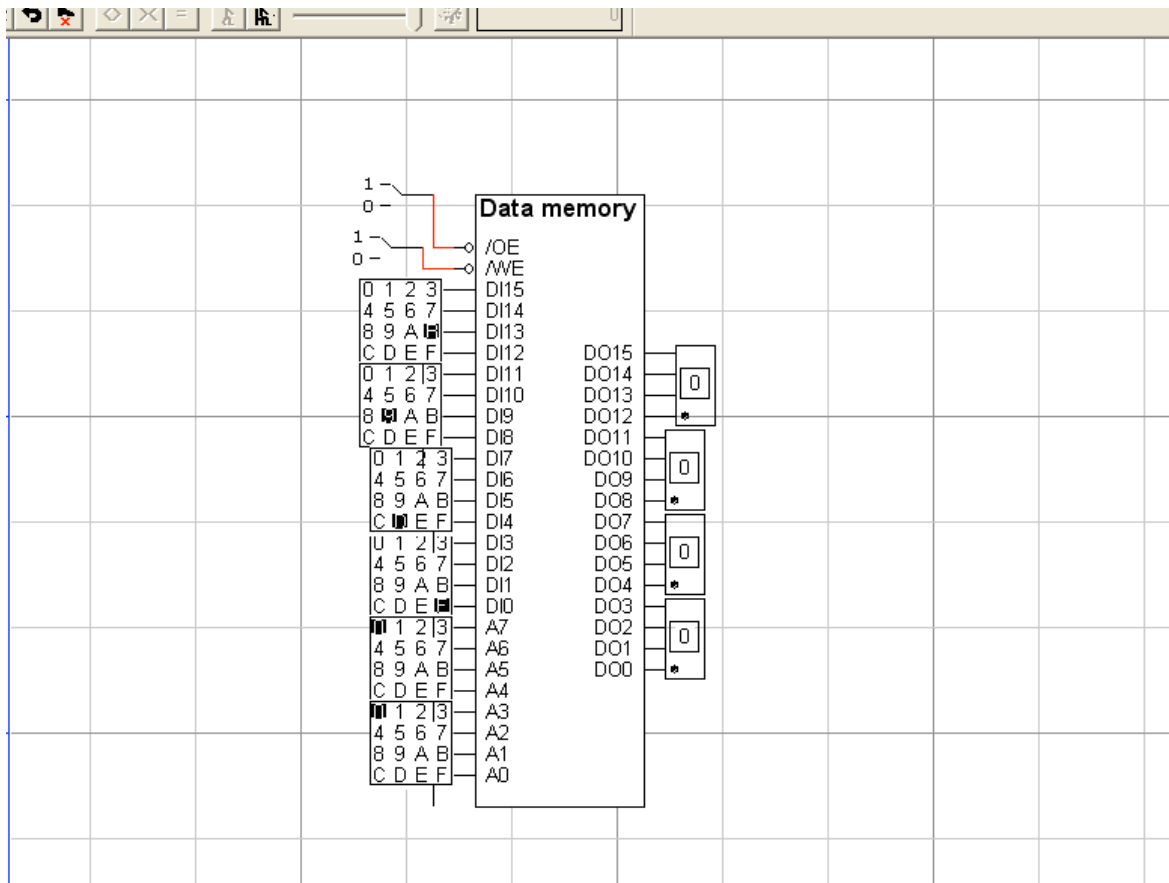
3. How many total bits of information can be stored with this device?

$16 \times 4 = 64$ bits

Connect the chip on your Protoboard so that you can write to and read from/display the contents of the memory locations.

Demonstrate to the instructor that you can successfully write to and read from the memory locations on the device.

Exercise 4: Open the *sram.cct* file:



How many addresses are there for this device?

8 address bits, so $2^8 = 256$ addresses

How many bits are stored at each address?

16 bits of data at each address

The **/OE** (Output Enable) line must be active to read the contents of the specified address (**/OE** is active low, or set to 0, when a read occurs).

The **/WE** (Write Enable) line must be active to write a new value to the specified (**/WE** is also active low).

Why do you think you need these two separate control signals? If we have multiple chips, we need OE to control which one is being read. We always need to control when we are writing new data with a WE.

Write the following addresses with the indicated values by doing the following:

1. Set $\overline{\text{OE}} = 1$ and $\overline{\text{WE}} = 1$ (this is the default state, which means you are neither reading or writing to the device (the “remember” state for the memory)).
2. Set the address lines **A0..15** to the location.
3. Set the data in **DI0..15** to the value to be written.
4. Activate the $\overline{\text{WE}}$ input (set to 0 then back to 1).
5. Repeat steps 2 – 4 for each address being written to.

<u>Address</u>	<u>Data In</u>
00	A5A5
01	3C3C
02	7878
04	0123
08	4567
10	89AB
20	CDEF
40	0F0F
80	FFFF

To read the indicated addresses to verify that the desired values are stored there, do the following:

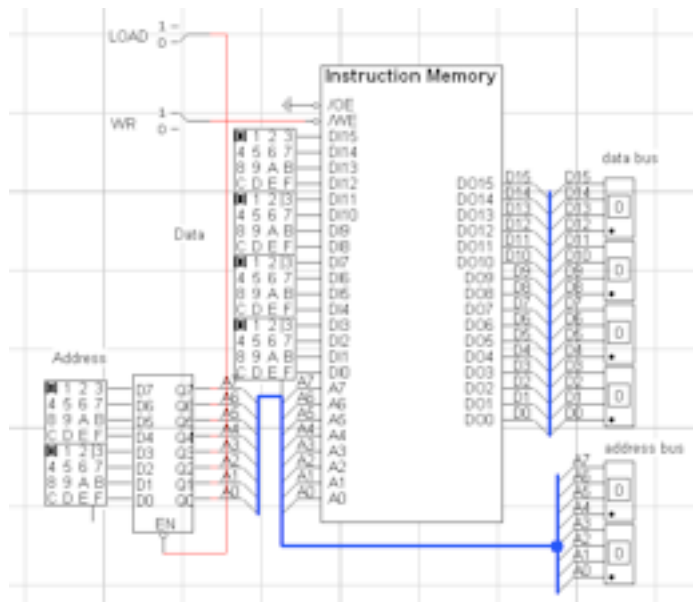
1. Set $\overline{\text{OE}}$ line to 0 (means enable the data outputs, so you can read the stored values)
2. Set address on **A0..15**, and verify that **DO0..15** reflect the values written in the earlier steps.
3. Repeat step 2 for each address.

Demonstrate to the instructor.

Memory Bus Circuit

Exercise 5 : This exercise will introduce you to a simple bus scheme which will be used in our implementation of the mini-MIPS architecture.

Open the *memory.cct* circuit from the Google group.



1. Use the memory chip in the *memory.cct* file for the **Instruction Memory**.

2. The device connecting the **Address** hex keyboard switches to the **Instruction Memory** is called a **Buffer-8 T.S.** in LogicWorks. It is a bank of 8 tri-state buffers, a device you have learned about in lecture

3. Use **breakouts** to make address and data line connections.. You can only connect two **breakout** together that have the same **breakout label**.

You can write values into memory locations by using the hex keypads on the left to specify the address and data values. The value of the currently selected address and its value are displayed on the hex displays on the right.

Follow the directions on the next page to test your circuit by writing to memory:

To write to memory (load a program):

1. Set **LOAD** to 0 (the **LOAD** line is activated by the user to load a new program to memory)
2. Set the **Address** switches to the location in memory at which you wish to write a data value.
3. Set the **Data** switches to the value you wish to write to that location.
4. Set **WR** to 0, then back to 1.
5. Repeat steps 2 - 4 for each address and its data value.

Using the **write** procedure, load the following data values to the specified addresses:

Addr	Data
00	5002

02	5003
04	1220
06	0230
08	2122
0A	8001

As you write the values, you will observe the address and data values on the data and address bus displays. At any time, you can observe the contents of an address by setting the address switches to the desired location.

Disconnect the address switches from the address bus by setting **LOAD** to 1. This will always be required when the user is finished loading the program into memory, and wishes to run the program. The address is supplied by the CPU during execution of the program, so the switches are not needed at that point (and would interfere with the CPU operation if still connected).