The National School of Artificial Intelligence
Computer Architecture
Computer Components

Lab 02: ROM and simulation

Objectives: ROM and simulation

In this lab, you will:

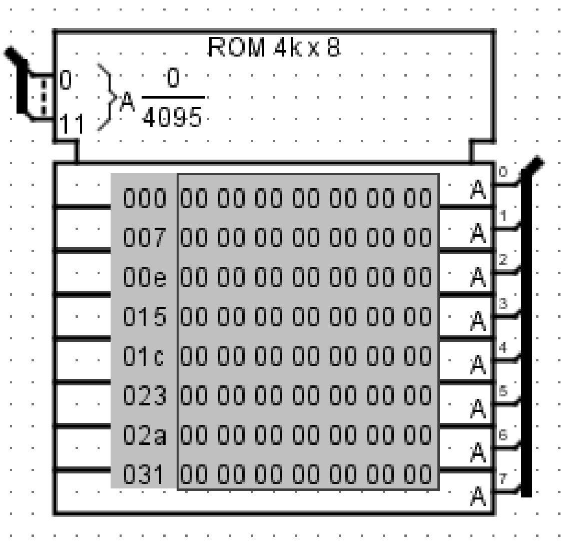
1. Build a fun application, *Magic 8-Ball*, using a ROM and test it using a clock (60 minutes);
2. Complete the assignment using the same as the example (30 minutes).

Try to do it yourself. If you find any problem ask your lab teacher.

Magic 8-Ball game

This lab introduces students to ROM and builds a fun application: The *Magic 8-Ball*. This was a toy that was developed in the 1950s and was popular throughout the 1960s. It was a small plastic sphere with the markings of an 8-ball. If the user "asked it a question" and then turned the toy upside down the answer would magically appear in a small window on the bottom of the ball.

Start a new Logisim Evolution project and create a subcircuit named Magic8Ball. Open that circuit and place a ROM (*Memory* library) device near the center of the drawing canvas. Set the ROM properties for an *Address Bit Width* of 12 and a *Data Bit Width* of 8.



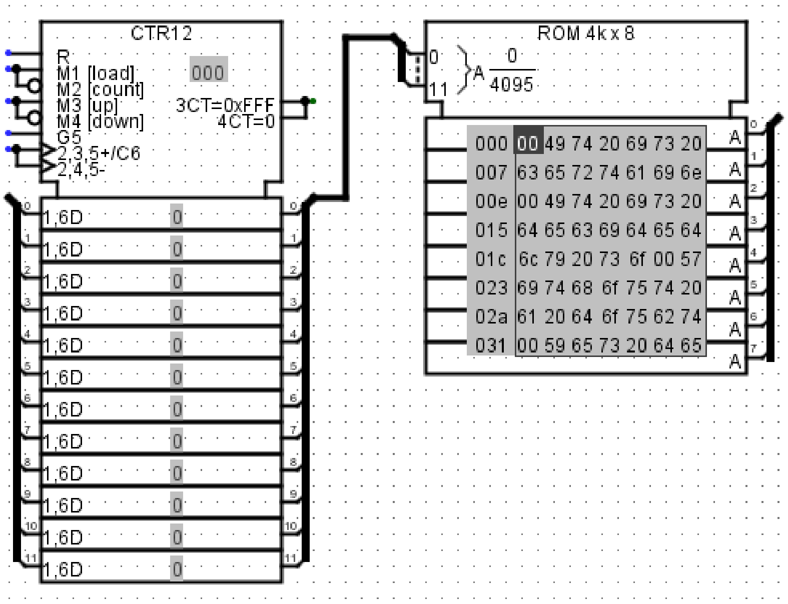
```
000 00 00 00 00 00 00 00 00
007 00 00 00 00 00 00 00 00
00e 00 00 00 00 00 00 00 00
015 00 00 00 00 00 00 00 00
01c 00 00 00 00 00 00 00 00
023 00 00 00 00 00 00 00 00
02a 00 00 00 00 00 00 00 00
031 00 00 00 00 00 00 00 00
```

Figure 1: Placing ROM

A ROM stores data that is accessed by setting an address on the inputs at the top left of the device and then reading the contents of that address on the 8-bit bus on the right side of the device. By attaching a counter to the ROM address port several consecutive addresses can be "stepped through" to output a message. Attach a Counter (*Memory* library) with 12 Data Bits to the address port of the ROM, as in Figure 2.

According to Wikipedia[1], the Magic 8-Ball featured 20 sayings:

```
 1 001 It is certain
 2 00f It is decidedly so
 3 022 Without a doubt
 4 032 Yes definitely
 5 041 You may rely on it
 6 054 As I see it yes
 7 064 Most likely
 8 070 Outlook good
 9 07d Yes
10 081 Signs point to yes
11 094 Reply hazy try again
12 0a9 Ask again later
13 0b8 Better not tell you now
14 0d1 Cannot predict now
15 0e4 Concentrate and ask again
16 0fe Do not count on it
17 111 My reply is no
18 120 My sources say no
19 132 Outlook not so good
20 146 Very doubtful
```



```
000 00 49 74 20 69 73 20
007 63 65 72 74 61 69 6e
00e 00 49 74 20 69 73 20
015 64 65 63 69 64 65 64
01c 6c 79 20 73 6f 00 57
023 69 74 68 6f 75 74 20
02a 61 20 64 6f 75 62 74
031 00 59 65 73 20 64 65
```

Figure 2: ROM With Counter

The *Magic 8-Ball* simulator built in this lab uses those same 20 saying. In the above chart, each saying is numbered and the start point in ROM (using hexadecimal notation) for each saying is also

[1] https://en.wikipedia.org/wiki/Magic_8-Ball

noted. Thus, saying one starts on ROM byte 001, saying two starts on ROM byte 00f, saying three starts on ROM byte 022, and so forth.

The content of the ROM device must be loaded before it can be used and that content is provided in `Lab_02_ROM.txt` accompanying this lab. To load the ROM device, click it one time and then click the "(click to edit)" link in its properties panel. In the ROM editor window that pops up, click the "open" button and navigate to the ROM memory file. Click "close window" to load the ROM device and make it ready for service.

The start point for each saying, as indicated on the above figure, is stored in a Constant (*Wiring* library) then a Mux (*Plexers* library) with five select bits is used to transmit a message start location to the counter so it can be read from the ROM device. Both constants and the multiplexer have a Data Bits of 12. The value of each constant should be set as described in the figure below. You can select an already set constant and copy-paste it, then change its value. Figure 3 illustrates the circuit at this point.

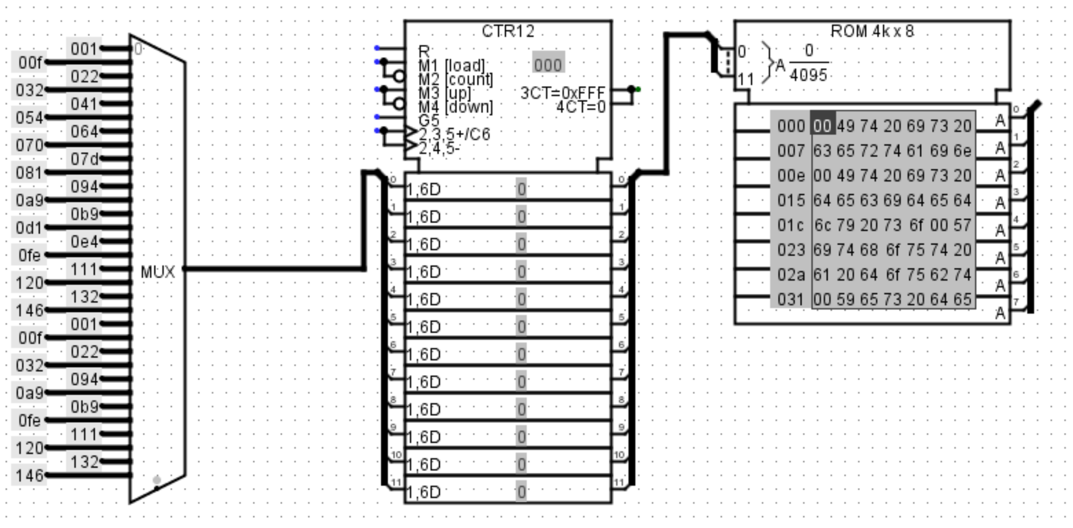


Figure 3: ROM Filter Mux

A five-bit Random Generator (*Memory* library) is used to select a random message. Figure 4 illustrates the placement of the random generator.

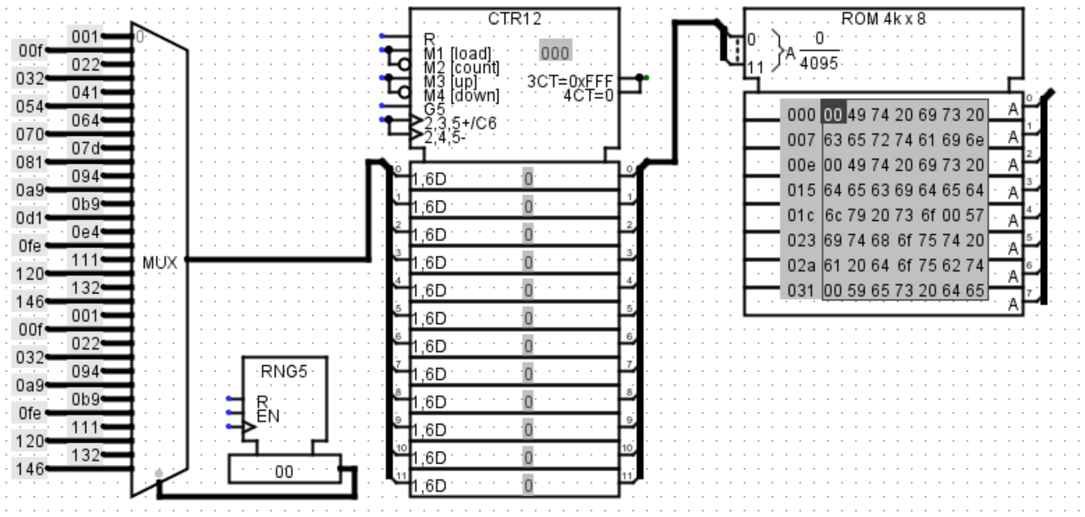


Figure 4: Random Generator Added

To complete the circuit, a few odds-and-ends were added. Figure 5 shows the completed circuit, but details from that figure are used below to describe how to complete the circuit.

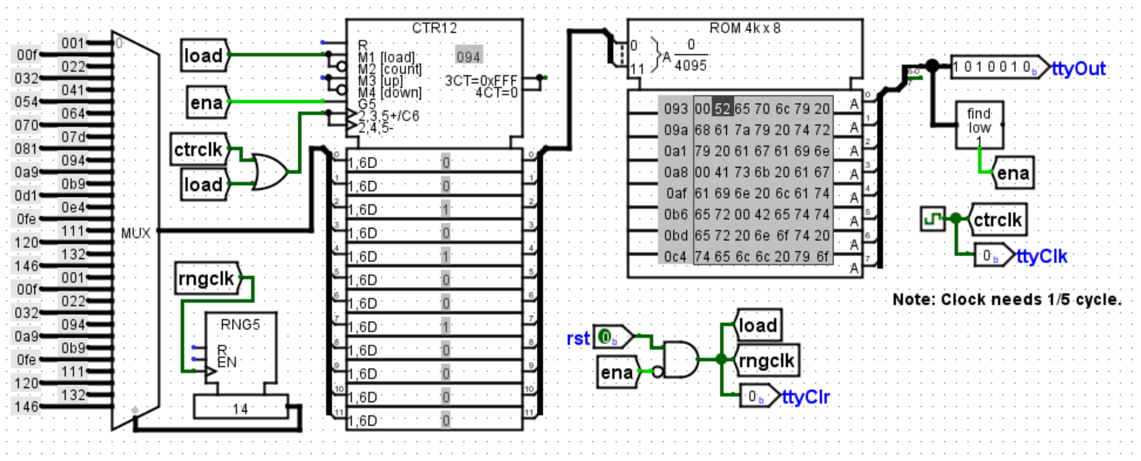


Figure 5: Completed Magic 8-Ball Circuit

To set up the counter, four signals are needed. These are all from tunnels (*Wiring* library) connected to other spots on the circuit. (See Figure 6). A tunnel is just a hidden wire between two or several components. We connect tunnels by giving them the same name (Label property). Thus, when a tunnel receives a signal it will be broadcasted to all tunnels with the same label. You can change the pointing direction of a tunnel by changing its facing property (East, West...etc.), or with the direction arrows on your keyboard. **Don't worry if the wires between the tunnels and the other components appear in blue. Blue means not correctly connected, thus we need to connect all the tunnels**.

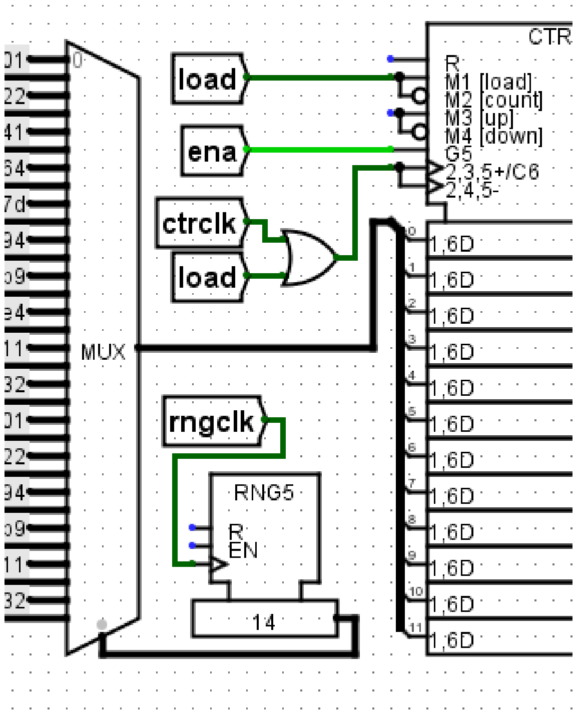


Figure 6: Counter Inputs

- **load**. The load signal goes high when the counter should be loaded with a new number from

the multiplexer. The number loaded is the location in ROM for the start of a message. Notice that the load signal is used on two pins. The top pin places the counter in load mode while the bottom pin uses the load signal as a clock pulse.
- **ena**. The enable signal turns the counter on/off. When enable is high then the counter functions normally and when it is low the counter is disabled.
- **ctrclk**.The counter clock provides the clock signal for the counter.

Connect the random number generator as follows. (See Figure 6.)
- The clock input pin is connected to a “rngclk” tunnel.
- The generator output is wired to the select port of the multiplexer.

The counter’s control signals are generated and distributed from a small group located under the ROM device. The purpose of this tiny group is to transmit a high signal through the AND gate when the reset pin goes high while enable is low. This generates the signals needed to select a new random message and put the starting address of that message in the counter. (See Figure 7.)
- **rst**: The reset pin is an external signal that originates from the main circuit. It is a simple 1-bit input pin (see toolbar) with “rst” as a label.
- **ena**: Enable Not originates from the ROM output group. The small circle is a NOT gate. It may look different on your version.
- **load**. This signal is used to load a message starting address into the counter. When it goes high it activates the “load” function and also becomes a single clock pulse for the counter.
- **rngclk**: The random number generator clock signal activates that device so it generates a random number. That number is then used to select a single line from the multiplexer so a message starting address can be loaded.
- **ttyClr**: Is a output pin with “ttyClr” as its label. This sends a high signal to the TTY “clear” pin on the main circuit. That signal is used to clear the TTY device. A TTY is a display screen that we use later.

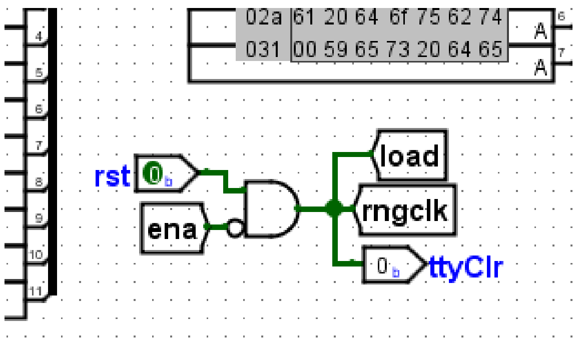


Figure 7: Counter Control Generation and Distribution

There are two functions found at the ROM device output. (See Figure 8.)
- The output of the ROM device is connected to the *ttyOut* pin in order to drive the teletype device on the main circuit. Note that the output of the ROM device is a splitter. The splitter creates a correspondence between a multi-bit value and several separate subsets of those bits. Despite its name, it can either split a multi-bit value into component parts, or it can combine component parts into a multi-bit value.
  Connect the splitter to the ROM’s output, change its "Bit Width In" property to 8, and leave "Fan Out" to 2. Eight rows were added at the bottom of the properties tab (Bit 0 to Bit 7). We can choose where we want those to be outputted. Since ASCII letters are only seven bits wide, this splitter passes bits 0-6 to the *ttyOut* port, but bit 7 (the most significant bit) is simply discarded. We do that by setting the value of bits 0 to 6 to Top.
- The **Bit Finder** (*Arithmetic* library) attached to the output of the ROM device is used to find

the lowest-order *one* in the ROM byte output. If the ROM byte includes at least one *one* then the south port of the finder is high. If the ROM output is all zeros then the Finder output goes low and that is used as the *ena* signal for the counter and random number generator. When the enable signal is low it also permits a *rst* signal (generated on the main circuit when the user "asks another question") to create a new answer. Change its "Data Bits" property to 7.

- The ttyOut is an output pin with 7 as its "Data Bits" property.
- Near the output of the ROM device a clock signal is split into two outputs. One is the *ctrclk* tunnel that is used by the counter and the other is the *ttyClk* pin, which is used on the main circuit to clock the teletype device. The clock can be found in the "Wiring" library. **It is important to note that the clock properties are set for a 1-tick high duration and 5 ticks low duration (a 1/5 clock).**

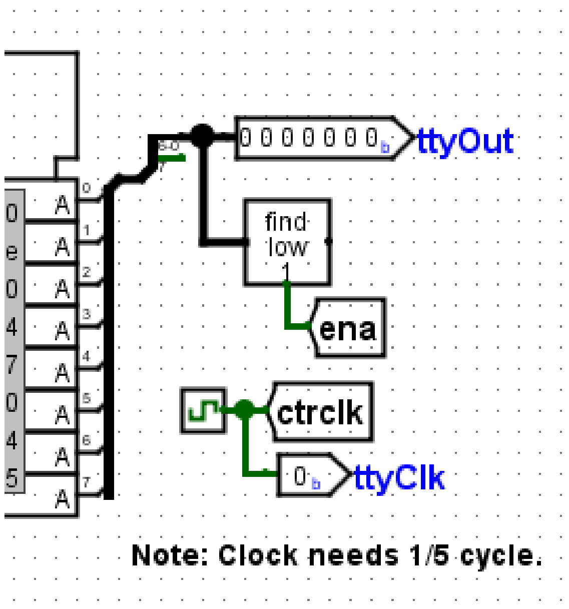


Figure 8: ROM Output

The only remaining step is to create the main circuit. As in all labs, the main circuit does nothing more than provide a user interface for the *Magic 8-Ball* Circuit. Figure 9 illustrates the main circuit.

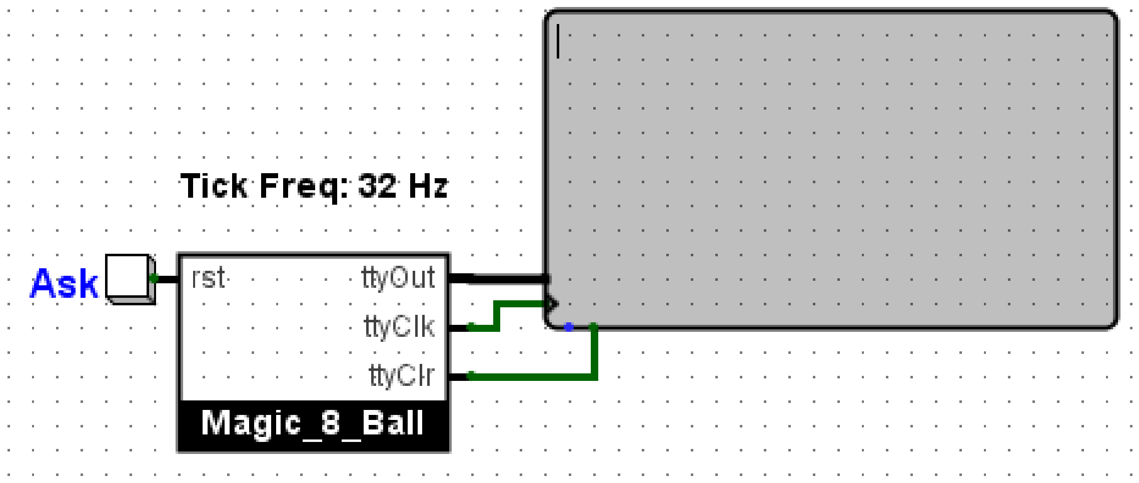


Figure 9: Magic 8-Ball Main Circuit

In main circuit, put the *Magic8Ball* circuit. Then, add a button, from the "Input/Outpt" library, with "Ask" as its label. Connect it to the "rst" input of our *Magic8Ball* circuit. Finally, add a "TTY" screen from the same library and connect it to the *Magic8Ball* circuit as shown in figure 9.

Testing the Circuit

Before the circuit can be tested the ROM device must be loaded. The ROM was loaded earlier in the lab but in case it does not have any content (it is filled with zeros), then load it with `Lab_02_ROM.txt`, which was provided with the lab. To load the ROM device, click it one time and then click the "(click to edit)" link in its properties panel. In the ROM editor window that pops up, click the "open" button and find the ROM memory file. Click "close window" to load the ROM device and make it ready for service.

The circuit should be tested by enabling the simulator clock at a frequency of 32 Hertz. To do that, go to the Simulate menu then Auto-Tick Frequency choose 32 hz. In the same menu click on Auto-Tick Enabled or Ctrl+K. Every time the *Ask* button is pressed a new random message will be displayed on the teletype screen.

Now, ask this magic ball the following question. Are going to have good grades in computer architecture? What's the answer?

Assignment

Using the same idea, build a circuit to display the following:

My name is: "your full name"
****
I am: "your age" years old
+++++++++++++++++++++++++
I got "grade" in Database.
____
I got "grade" in WebDev.
/////////////////////////
I got "grade" in Math.
****
I got "grade" in the other math module.

You have to put your full name, your age, and your grades. Use any online converter to convert the text to hexadecimal.

Submit this circuit and the *Magic8Ball* circuit.