Carleton University

Department of Systems and Computer Engineering

SYSC 2310 Fall 2017

Introduction to Digital Systems

Lab #4

Lab Report Due: At **6:00 am** on the day that is one week after your scheduled lab session. (E.G. if your lab is on Tuesday, the report is due on Tuesday one week later.)

Submit the report as a single PDF file. Place the report in a folder named Lab4, along with your lab work as described below. Zip the folder (Lab4.zip) and submit it to the appropriate location in the cuLearn webpage for your lab section (not the main course webpage!).

Only submit your work in the Windows-compatible ZIP format. Do **not** submit your work in RAR, 7z or other format. (If the TAs can't open your work using Windows' built-in zip utility then they will not mark it.)

Note that the system will automatically stop accepting submissions at the due date/time for your lab section.

LATE submissions will NOT be accepted.

Version 2: includes a highlighted addition of Splitters to the list of acceptable components in Step 3.

Read this document completely and carefully before deciding what to do.

Please bring questions about the lab to class for discussion at the beginning of class.

In this lab you will:

- **Design a circuit that controls the movement of a "ball" on a "display" screen.**
- Enter circuit designs into Logisim.
- Test and debug circuits using Logisim's simulation capability.

"[Report]" indicates content that should be addressed in the lab report.

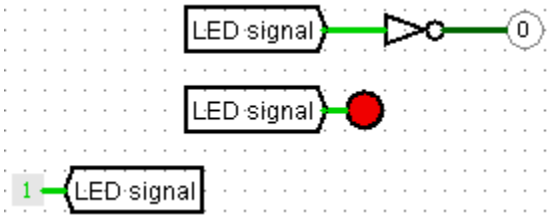
Step 1: Read this entire document before proceeding to Step 2.

This lab introduces the Tunnel and Clock features of Logisim. Both are located in the Wiring Library.

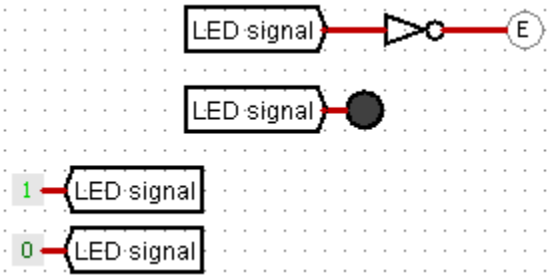
Tunnels: A Tunnel is a connector to an internal wire hidden inside Logisim. Each unique Tunnel Label creates a unique hidden internal wire. All Tunnels with the same Label are connected to the same hidden internal wire. This allows Tunnels to be placed at convenient locations in a circuit, and the wiring

to connect the Tunnels does not need to be drawn. Tunnels are a convenient way to avoid having to draw long wires, and also a handy way to provide convenient names for the signals being carried through the Tunnels.

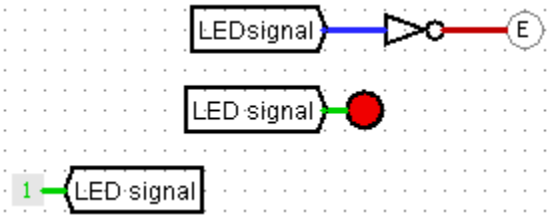
The circuit below shows a Constant 1 driving both an LED and an inverter. Instead of drawing wires to connect the components, the Constant is connected to a Tunnel Labelled “LED signal”, and this Tunnel is the source of a signal that is automatically connected (the connection is hidden inside Logisim) to any Tunnel with the same Label. A second Tunnel with the Label “LED signal” is connected to the LED, and the LED is ON (as would be expected if the Constant 1 were connected directly to the LED). A third Tunnel Label “LED signal” is connected to the Inverter, which is in turn connected to a Probe. The Probe shows that inverting LED signal gives 0 (as would be expected if the Constant 1 were connected directly to the Inverter).



Tunnels are very convenient for organizing and de-cluttering circuit diagrams, but they have some aspects that require caution:



There is no limit on the number of Tunnels that can have the same Label (so a signal can be easily routed to many places in a circuit). The circuit above shows what happens if two different components attempt to drive conflicting values through Tunnels with the same Label. In this case, a constant 0 has also been connected to a Tunnel Labelled “LED signal”. Now there are two different sources (constant 0 and constant 1) attempting to drive the “LED signal”, and Logisim has generated errors (red wires!) since the LED signal can’t have two different values at the same time.

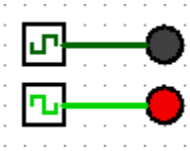


The typing of Tunnel Labels also requires care. The circuit above shows a Tunnel with a typo in the name. The Label on the Tunnel connected to the Inverter has a typo, and is “LEDsignal” (no space).

Logisim has assumed the miss-Labelled Tunnel is “just another” unique Tunnel and assumes it will be used elsewhere as the circuit is developed. Logisim is showing that the signal is undefined (blue wire!), since there is no Tunnel with the “LEDsignal” Label that is being driven to a value by some component. Furthermore, the Inverter output is an error (red wire) since Logisim can’t invert an undefined value.

The examples above use Tunnels to communicate a single signal. Each Tunnel has a Data Bits Attribute that can increase the width of the Tunnel. And of course, two Tunnels with the same Label but different widths will create “Incompatible Width” errors (orange wires!).

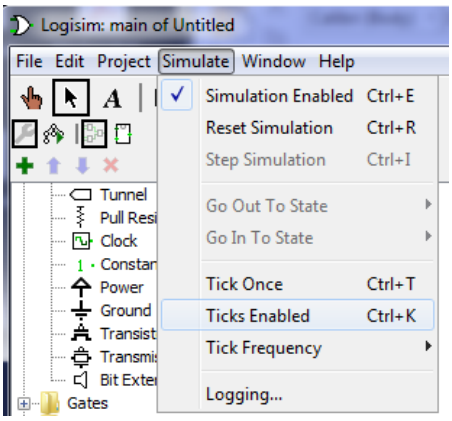
Clocks: A clock outputs a periodic signal (as discussed in class and the slides). A Logisim Clock oscillates between 0 and 1 at a constant frequency.



The circuit above shows two Clocks in different states. Note that the icons and wires are being explicit about the Clock’s current state.

If you open Logisim and try to reconstruct the above circuit hoping to see blinking LEDs, you will be disappointed since the circuit will just sit and stare back at you without oscillating. ☹

By default, Logisim’s Simulator disables Clock components. Clocks must be Enabled before they will oscillate in the Simulator. The figure below shows that Logisim’s Simulate menu has three entries related to Clocks: Tick Once, Ticks Enabled, and Tick Frequency. You must click the Ticks Enabled entry to turn Clock oscillation ON (or OFF). When Ticks are Enabled, a check mark will appear beside the entry (similar to the check mark beside the Simulation Enabled entry).The Menu entry Enables (or Disables) all of the Clock signals at the same time. The “Tick Once” entry can be used to advance the simulation by one Tick, but it actually advances by only one Clock signal change … i.e. a half a clock period. This can be extremely useful when testing and debugging your circuit! Sometimes opening Menus and selecting entries is a pain in the interface ☺, and Logisim provides hotkeys (Ctrl+T and Ctrl+K) for these Tick-related menu entries. These can be used directly from the keyboard without mousing-around with Menus (e.g. “Ctrl+T” means hold down the Ctrl key, and while it is down, press the T key).



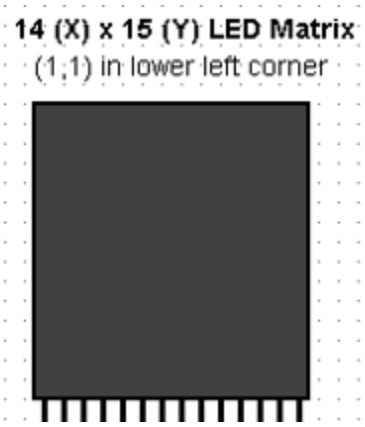
All Logisim Clock components in a circuit operate at the same clock Frequency. The Frequency may not be accurate in "real" time (i.e. wall-clock time), but Logisim "tries its best". The default frequency is 1 Hz. The Frequency (of all Clocks in a circuit) can be adjusted using the Tick Frequency entry (in the Simulate Menu show above).

Logisim's default Clock duty cycle is 50%, but that can be changed in each Clock's Attributes. There is no need to change the duty cycle in this lab.

Step 2: Download the Lab4.zip file to a persistent work area, and unzip the folder.

The file Lab4Start.circ in Lab4.zip has been provided as the starting point for the lab circuits. The Lab4Start circuit has a lot of components that are introduced and discussed below in "pieces" … perhaps it might be useful to open the circuit and read about the "pieces" in context of the circuit.

The lab involves "bouncing" a ball on a "display". The display is the Logisim LED Matrix shown below:



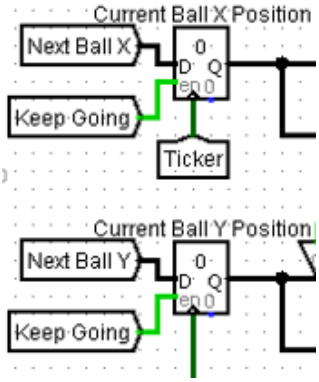
The display is a 2-dimensional array of 210 square LEDs. Each LED has an (X,Y) coordinate, with X measured along the horizontal (row) direction and Y measured along the vertical (column) direction. The dimensions of the array are 14 rows (X) by 15 columns (Y). The coordinates of the LED in the bottom left corner are (1,1) (i.e. row 1, column 1), and the coordinates of the LED in the upper right corner are (14,15).

The display will be manipulated to give the impression of a ball moving in time. Assume that the outside edges of the display are solid walls, so when the ball reaches an edge it will bounce back into the display rather than just fall off the display.

The "ball" is represented using a single LED on the display. The current position of the ball is shown by turning ON the single LED at the current position of the ball. Movement of the ball can be shown by turning ON/OFF different LEDs over time. Only one LED must ever be ON at one time, and that LED indicates the ball's current position.

In reality, the displays on your phones, TV and monitors are just fancier versions of this very simple ON/OFF (0/1) "display" approach (they have much smaller coloured LEDs, called pixels). Graphics programming is really about manipulating pixels (0's and 1's)!

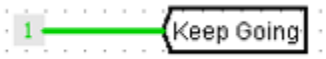
The ball's current position is "remembered" in 2 Registers with the Labels: Current Ball X Position and Current Ball Y Position. These are 4-bit Registers because the X position will range from 1 to 14, therefore requires 4 bits to encode), and the Y position will range from 1 to 15 (also requires 4 bits to encode). These registers are shown below:



The ball's current position is stored in these Registers. The Tunnels attached to the Registers' inputs are providing the signals used by the Registers.

As the ball moves, the next position (Next Ball X and Next Ball Y signals) must be calculated and (eventually) saved. The circuits in Step 3 and 4 calculate and output the Next Ball X Position and Next Ball Y Position signals to the Registers.

The Keep Going signal is input to the Registers' "enable" control and is used later in Step 5 (and will be discussed in more detail then). For now, notice that the Keep Going signal is held at the Constant value 1 (don't change this until Step 5):



A Clock is used to move the ball at a consistent speed on the display. The ball's position should change every Clock cycle (except later in Step 5 ☺). The Clock signal is connected directly to the to the Current Ball Y Position Register's edge sensitive control input, and is communicated to the Current Ball X Position Register using a Tunnel Labelled "Ticker". The source of the Ticker signal is shown below.

Each Clock cycle of Ticker will be referred to as a Tick. Since the Ticker signal is input to the edge sensitive control inputs on the Registers, a new ball position (Next Ball X and Next Ball Y) will be saved every Tick. The ball's position will change every Tick to give the visual impression of the ball moving on the display.

If you Enable Ticks (see above) you will see the periodic changes in the Ticker signal … but there is no need to enable the Clock yet, and the constant changes can be annoying (visually).

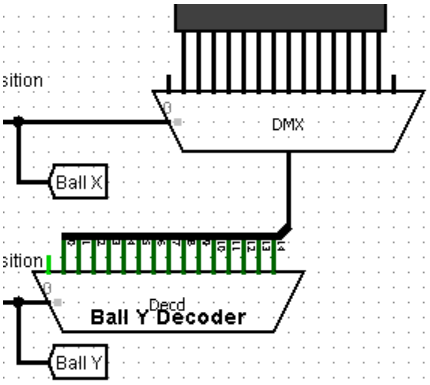
The LED Matrix has 14 inputs, with one for each column. The ball's current X position determines which column contains the LED associated with the ball. The ball's current Y position determines which LED in the column should be turned ON. Each column input is 15 bits wide, with one bit per LED in the column. The least significant bit of the 15-bit column input is associated with the LED at the bottom of the column (Y = 1), and the most significant bit is associated with the LED at the top of the column. An LED can be turned ON by inputting a 1 to its associated bit in the 15-bit column input. More than 1 LED can be turned ON in a column at once, but this lab will only have one LED turned ON at a time. All of the 210 LEDs must be controlled ON or OFF at once (i.e. they must have either a 0 or 1 driving them)! How to control this is discussed soon, but first explore the display by positioning the ball at various locations.

To see the relationship between the Current Ball X Position/Current Ball Y Position and the display, Poke some values into the Registers. First Poke 1 into the Current Ball X Position Register and Poke 1 into Current Ball Y Position Register. The ball should appear at (X,Y) = (1,1) on the display (i.e. at the lower left corner).

Now Poke 7 into the Current Ball Y Position Register. The ball now appears at (X,Y) = (1,7) on the display (i.e. around the middle of the first column).

Now Poke 8 into the Current Ball X Position Register. The ball now appears at (X,Y) = (8,7) on the display (i.e. around the middle of the display).

The control of the LED Matrix is managed by the Multiplexer (for X) and the Decoder (for Y) shown below. The values output from the Current Position Registers are communicated to other sub circuits through Tunnels Labelled “Ball X” and “Ball Y”.



The Ball Y Decoder is used to generate the 15-bit column input pattern to indicate the ball’s current Y position (recall the discussion above about the 15-bit inputs). The Decoder outputs a 1 at the bit associated with the ball’s current Y position, and outputs 0 for all other bits in the column. Poke a non-zero value into the Current Ball Y Position Register and see which Decoder output goes to 1.

The Ball X value is used select the display column in which the ball will appear. The Ball X signal is used as the control inputs on the 16-to-1 MUX. Only 14 of the MUX outputs are used, since there are only 14 columns on the display. The MUX routes the 15-bit pattern generated by the Ball Y Decoder to the selected display column. The MUX sends a 15-bit pattern = all 0’s to the columns that are not selected. As a result, the MUX outputs 210 bits at all times, and all of the bits are 0 except for the bit associated with the ball’s current position. ☺ The MUX outputs meet the requirements for showing the ball on the display.

The Stuck, Reset and Game Over sub circuits are used later in lab (and will be discussed then). For now, ignore these and leave them as currently configured until a Step says to modify them.

Step 3) Bounce in X Direction: Extend Lab4Start to “bounce” the ball in the X direction. The ball should start at (X,Y) = (1,7). On every rising edge of Ticker, the ball should move one column (X) to the right. When the Ball reaches the right side of the display (column 14), the ball should “bounce” off the side of the display and move back towards the left side of the display (at the same speed). Once the ball reaches the left side of the display (column 1) it should “bounce” off the side of the display and move

back towards the right side of the display. The bouncing should continue indefinitely. The ball should appear in each column of the display for one Tick.

Design your solution around the use of the "Register plus adder" counter presented in the Memory slides and presented in class (see discussion of Counters in the slides). The design should count the rising edges of Ticker (Recall that Lab4Start is already configured to trigger the Ball registers on the rising edges of Ticker). Your version of the counter will take Ball X as an input and generate the Next Ball X value as the output. Counting "up" (by adding 1) will only work while the Ball is moving towards the right side of the display. Your counter will also have to count "down" (by subtracting 1) while the Ball is moving towards the left side of the display!

Use the variation on an adder circuit (in the Information Encoding slides) to either count up or down. Hint: You will also need something to remember which way you are counting until the next "bounce". ☺

Hint: Comparators might be very useful here (and elsewhere in the Lab), but be sure to have the Attributes set properly!

Reminder: the ball won't move while Ticks are disabled.

Your design for this Step may use only the following Logisim components:

One (4-bit) Adder

Comparators, D Flip Flops, Logic Gates, Tunnels, wires, Splitters and Constants

Save your solution in Lab4X.circ in the Lab4 folder.

[Report] State whether your Lab4X circuit works successfully, i.e. it performs as required. The TAs will only consider a circuit that you have explicitly identified as working successfully. Marks will be deducted for claiming that a submitted circuit works when in fact it doesn't, so only submit a working circuit.

[Report] Describe how your circuit works to implement the requirements (don't duplicate the descriptions given in Step 2, and when appropriate, use the terminology used in Step 2). Try to be clear in your explanation. Be sure you have described the inputs, the outputs, and the purpose and operation of each sub circuit you have designed. Introducing Tunnels with convenient signal names into your circuit may help in your descriptions of the circuit. If it helps to clarify your descriptions, use visual snippets as done in Step 2 (the Windows Snipping Tool is quite easy to use). The task of constructing a clear and concise description is not trivial, be sure to spend an appropriate amount of time, as most of the Report marks are based on this description.

The TAs will grade your description in general categories. For example, if the description is worth 10 marks the categories might be: Design works and description is clear and concise (10 marks), Design works and description good but some room for improvement (7 marks), Design works but description has significant flaws or can't be read and understood easily (3 marks), Design works and the description is either missing or too convoluted to comprehend (1 mark).

Step 4) Bounce in Y Direction:

Make the ball bounce in the Y direction as well as the X direction by extending your Lab4X.circ with a sub circuit that is almost identical to what was built in Step 3.

Hint: Copy and Paste is your friend, but remember the earlier warning about getting the Tunnel Labels right!

Once the ball is bouncing in both directions successfully, save your solution in Lab4XY.circ in the Lab4 folder.

[Report] State whether your circuit works successfully, i.e. it performs as required. The TAs will only consider a circuit that you have explicitly identified as working successfully. Marks will be deducted for claiming that a submitted circuit works when in fact it doesn't, so only submit a working circuit.

Step 5) Add a Sticky Region to the Display:

Suppose that a box-shaped region (area) of the display is "sticky" and when the ball is in that region it moves at half speed. The corners of the sticky region are at (4,4), (4,12), (12, 12) and (12,4). The region also includes the columns and rows between the corners, e.g. (4,4) through (4,12) are in the sticky region.

Extend Lab4XY.circ (as discussed below) to implement the half-speed behaviour when the ball is inside the sticky region and save your solution in LAB4Sticky.circ in the Lab4 folder.

It is bad practice to put a Clock signal (like Ticker) through additional logic gates (for reasons we will discuss in class later). Instead of modifying the Ticker circuit, use the Keep Going enable input on the Ball registers to disable the Registers for every other Tick (Hint: the slides discuss how to build a circuit that divides a clock frequency in half.) The following steps should help you in developing your design.

Stuck Detector: Start by designing a sub circuit that turns on the Stuck LED (included in Lab4Start) whenever the ball is in the sticky region. The circuit should have one output, and that output should drive a Tunnel with the Label "Stuck". In order for this to work (and avoid errors in your circuit), the Constant 0 that has been driving the Stuck LED in the above steps must be disconnected from its Stuck Tunnel. Test your circuit to see that the LED is working properly as the ball moves in/out of the sticky region.

Keep Going: Now disconnect the Keep Going Tunnel from the Constant 1 that has been driving it for the above steps. Design and implement a sub circuit that will generate (output) the Keep Going signal. The circuit should have one output, and that output is connected to (i.e. drives) a Tunnel that has the Label "Keep Going" (Logisim will automatically connect it to the Registers via the existing Keep Going Tunnels). Your circuit will need to use the output of the Stuck Detector to help in deciding when the Keep Going signal should be 1. While the ball is in the sticky region, Keep Going should be 1 during every other Tick (Hint: if you are having trouble getting started here, analyse all of the conditions under which Keep Going should be 1 or 0, and then design a circuit to implement those conditions.)

Your design for this Step may use only the following Logisim components:

Comparators, D Flip Flops, Logic Gates, Tunnels, wires, and Constants

Test your circuit. When it works, save your solution in LAB4Sticky.circ in the Lab4 folder.

[Report] State whether your circuit works successfully, i.e. it performs as required. The TAs will only consider a circuit that you have explicitly identified as working successfully. Marks will be deducted for claiming that a submitted circuit works when in fact it doesn't, so only submit a working circuit.

Step 6) Reset Logic:

Extend LAB4Sticky.circ to have the following Reset Logic and save your solution in Lab4Reset.circ.

When the Reset button (included in Lab4Start) is pressed, the ball should start moving from the Reset Coordinates: (X,Y) = (1,7). To accomplish this, include some Reset logic that extends your circuits from Steps 3 and 4. The logic should introduce the Reset Coordinates between the current circuits and their output tunnels (Next Ball X and Next Ball Y) (e.g. for the Step 3 circuit: disconnect the Next Ball X output tunnel, add some reset logic, and connect the Next Ball X tunnel to the output of the Reset logic)

Some Reset logic will also be needed to extend your Step 5 sub circuit.

Your solution to this Step must not introduce any new flip flops.

Test your circuit. When it works, save your solution in LAB4Reset.circ in the Lab4 folder.

[Report] State whether your circuit works successfully, i.e. it performs as required. The TAs will only consider a circuit that you have explicitly identified as working successfully. Marks will be deducted for claiming that a submitted circuit works when in fact it doesn't, so only submit a working circuit.

Bonus) Game Over:

Extend the LAB4Reset.circ to have the following Game Over logic and save your solution in Lab4Bonus.circ in the Lab4 folder.

Assume there is a "hole" in the bottom wall of the display. The hole is a strip that is one LED wide that extends from (X,Y) = (5,1) to (7,1), inclusively. When the ball enters the hole, it should appear for one Tick (as before) and then disappear from the screen (and not reappear), and the Game Over LED should turn on. The Game Over LED is included in Lab4Start.circ.

[Report] State whether your circuit works successfully, i.e. it performs as required. The TAs will only consider a circuit that you have explicitly identified as working successfully. Marks will be deducted for claiming that a submitted circuit works when in fact it doesn't, so only submit a working circuit.

[Report] Briefly describe (in high level terms) how your solution "stops" the bouncing ball behaviour.

The lab is done! ☺ If you could not complete the lab during the scheduled lab time … that's OK … finish up the lab on your own time.

Before you leave the lab (even if you could not complete the lab), show the TA what you have completed. [Attendance marks are associated with this.]

Before the due date for your lab section (i.e. does not have to be done during the scheduled lab): Prepare your report and place the single pdf in the unzipped Lab4 folder containing your lab work. Name the report Lab4Report.PDF (the folder should retain the name Lab4). Zip and submit the Lab4.ZIP folder to the appropriate place in your lab section's cuLearn page.