A white background with black text

Description automatically generated

Department of Computer Science

Institute of Business Administration, Karachi

Abdullah Iqbal 26904

Lab #7: Logisim-Evolution, Sequential Circuit, and Flipflop.

Digital Logic Design

# Objective:

The objectives of this experiment are:

1. Get familiar with Logisim-evolution.
2. Use Logisim-evolution to model and simulate the behavior of digital logic circuits.
3. Understanding the concept of sequential circuit
4. Designing different types of flip flop.

# Introduction:

*Logisim-Evolution* is a Java application, so a Java runtime environment will need to be installed before using the application. Many students who are taking a digital logic class already have a Java runtime on their computer and can skip this step, but those who do not will need to install the Java runtime. That process is not covered in this man- ual but information about installing the Java runtime environment is available at <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. It can be confusing to know which version of Java to download but students working on the labs in this manual only need the runtime, called *JRE* on the website. Students who are also in programming classes will likely already have the runtime as part of the Java Developer’s Kit (JDK). It can be tricky testing the Java installation since the Chrome, Firefox, and Edge browsers will not run Java apps, but students can open a command prompt and enter java -version to see what version of Java their computers are running, if any.

*Logisim-Evolution (*[logisim-evolution download | SourceForge.net](https://sourceforge.net/projects/logisimevolution/)) is available as a free download. Visit the website and about halfway down the page find a section named “Running Logisim-evolution.” Click the “here” link at the end of the first sentence in that section.

Since the *Logisim-Evolution* file is a Java application, it does not need to be installed like most software. To start *Logisim-Evolution,* double- click the *Logisim-Evolution* shortcut. That will start Java and then run the *Logisim-Evolution* application. Also, *Logisim-Evolution* will not need to be uninstalled when it is no longer needed since it is not actually installed, the *Logisim-Evolution* file can simply be deleted.

* + 1. *Beginner’s Tutorial*

*Logisim-Evolution* comes with a beginner’s tutorial available in Help

-> Tutorial. That tutorial only takes a few minutes and introduces.

students to the major components of the application. Students should complete that tutorial before starting this lab.

* + 1. *Logisim-evolution Workspace*

Start *Logisim-Evolution* by double-clicking its icon. The initial *Logisim- Evolution* window will be similar to Figure [1.1](#_bookmark18).

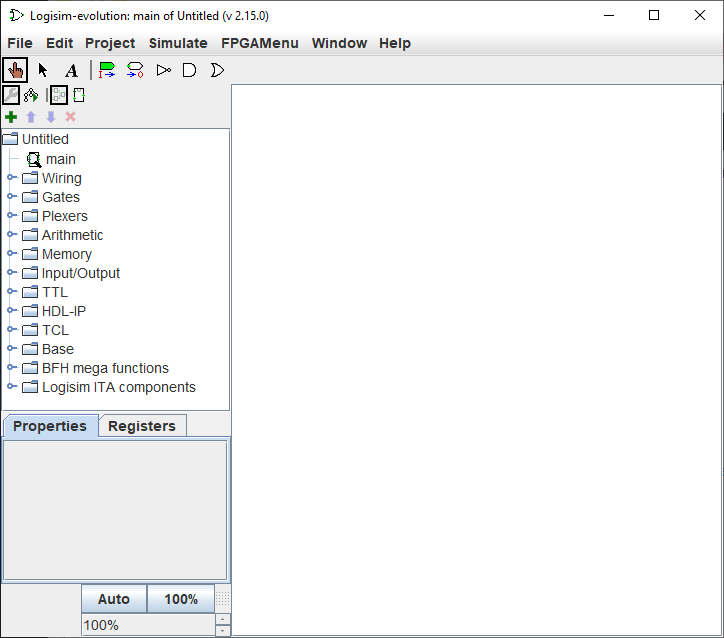


Figure 1.1: Logisim-evolution Initial Screen

The *Logisim-Evolution* space is divided into several areas. Along the top is a text menu that includes the types of selections found in most programs. For example, the “File” menu includes items like “Save” and “Exit.” The “Edit” menu includes an “Undo” option that is useful. In later labs, the various options under “Project” and “Simulate” will be described and used. Items in the “FPGAMenu” are beyond the scope of this class and will not be used. Of particular importance at this point is “Library Reference” in the “Help” menu. It contains information about every logical device available in *Logisim-Evolution* and is very useful while using those components in new circuits.

Under the menu bar is the Toolbar, which is a row of eight buttons that are the most commonly used tools in *Logisim-Evolution* :

* + - * **Pointing Finger**: Used to “poke” and change input values while the simulator is running.
* **Arrow**: Used to select components or wires in order to modify, move, or delete them.
* **A**: Activates the Text tool so text information can be added to the circuit.
* **Green Input Port**: Creates an input port for a circuit.
* **White Output Port**: Creates an output port for a circuit.
* **NOT Gate**: Creates a NOT gate.
* **AND Gate**: Creates an AND gate.
* **OR Gate**: Creates an OR gate.

The Explorer Pane is on the left side of the workspace and contains a folder list. The folders contain “libraries” of components organized in a logical manner. For example, the “Gates” folder contains vari- ous gates (AND, OR, XOR, etc.) that can be used in a circuit. The four icons across the top of the Explorer Pane are used for advanced operations and will be covered as they are needed.

The Properties panel on the lower left side of the screen is where the properties for any selected component can be read and set. For ex- ample, the number of inputs for an AND gate can be set to a specific number.

The drawing canvas is the largest part of the screen. It is where circuits are constructed and simulated.

* + 1. *Simple Multiplexer*

A multiplexer is used to select which of two or more inputs will be connected to a single output. For this lab, a simple two-input, one- bit multiplexer will be built. It is understood that students will not know the significance of a multiplexer at this point in the class, but the purpose of this lab is to use *Logisim-Evolution* to build a simple circuit and a multiplexer serves that purpose well.

Start by clicking the *And* button on the toolbar and placing two AND

gates on the canvas. The canvas should resemble Figure [1.2](#_bookmark20)

A black and red dotted letter

Description automatically generated

Figure 1.2: Two AND Gates

*Do not be concerned with the exact placement of components on the drawing canvas.*

*They can be rearranged as the building progresses.*

Click one of the AND gates to select it and observe the various properties available for that gate, as seen in Figure [1.3](#_bookmark21). The default values do not need to be changed for this circuit; however, all circuits in this manual use the “Narrow” gate size in order to make the circuit fit the screen better. The other properties will be explained as they are needed.

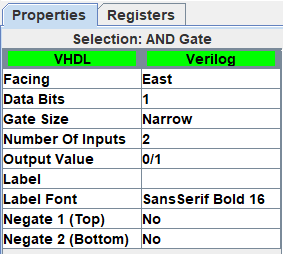


Figure 1.3: AND Gate Properties

The outputs of the two AND gates need to be combined with an OR

gate. Add an OR gate as illustrated in Figure [1.4](#_bookmark22).

A black and red dotted line with blue dots

Description automatically generated with medium confidence

Figure 1.4: OR Gate Added to Circuit

The top input for the first AND gate needs two NOT gates (inverters) so the two AND gates can function as on/off switches. This is a rather common digital logic construct and when the circuit is complete it will become clear how the switching function works.

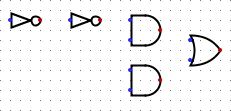


Figure 1.5: Two NOT Gates Added to Circuit

All inputs and outputs need to be added as in Figure [1.6](#_bookmark24). Note: inputs are square and outputs are round. The *Label* property for each input and output should be specified as in the figure. The pins are labeled according to their function in the circuit. Pin *Sel* carries a signal that selects which input to connect to the output, pins *In1* and *In2* are the two inputs, and pin *Out1* is the output. Note: output pins display a blue-colored X until they are actually wired to some device like the OR gate in the illustration.

A black and red dotted lines with blue dots

Description automatically generated with medium confidence

Figure 1.6: Inputs and Output Added

Finally, connect each device with a wire by clicking on the various ports and dragging a wire to the next port. To start the wire in the middle of the two NOT gates click the wire connecting those gates and drag downward. Wires will automatically “bend” one time but to get two bends, like between the output of an AND gate and the input of the OR gate, click-and-drag the wire from the output of the AND gate to a spot a short distance in front of that same gate, then release the mouse button and then immediately click again to start a new wire that will “bend” to the input of the OR gate. Only a little practice is needed to master this wiring technique.

A diagram of a circuit

Description automatically generated

Figure 1.7: Circuit Wiring Added

To operate the circuit in a simulator, click the *Pointing Finger* and “poke” the various inputs. If it is working properly, when the *Sel* in- put is high then the value of *In2* should be transmitted to the output, but when *Sel* is low then the value of *In1* should be transmitted to the output. This circuit is used to select one of two inputs to be transmitted to the output.

* + 1. *Chronogram*

*Logisim-Evolution* can generate a timing diagram, called a *chronogram*, for a sequential circuit. That is a representation of the various signals in a circuit and how those signals change over time. Figure [1.8](#_bookmark107) is the timing diagram for an Up counter.

A screenshot of a computer screen

Description automatically generated

Figure 1.8: Timing Diagram for Up Counter

At the top of Figure 1.8 is a scale that indicates the number of seconds that the counter has been operating. The first trace is the input clk signal. The clock goes high at the start of each second and then goes low at the half-second mark. Under the clock is the “Probe1” signal. Because that is a four-bit number *Logisim-Evolution* displays the number, but under that number is a breakout of the four bits that make up that number. Thus, at time zero “Probe1” is 0001 and “Probe1\_s\_0” (that stands for “Probe 1, Signal 0”) is high while the other bits are low. The *Logisim-Evolution chronogram* includes a cursor indicated by a red line (found just before the five second tick in Figure 1.8) that can be placed anywhere along the diagram. The cursor sets the values of each signal in the area on the left edge of the diagram, so the cursor in Figure 1.8 is pointing to a spot where the *clk* is low, *Probe1* is at 0101, and so forth.

Follow the next steps to use the chronogram. Notes: the chrono- gram will only check subcircuits that are found on the main subcircuit. Therefore, in order to create a timing diagram all subcircuits need to be combined on main. The labs completed in this manual have been designed to use the main subcircuit as the human interface so the chronogram feature will work well with these circuits.

* + - 1. In the main subcircuit, add a “sampling clock” labeled *sysclk* (this name is important, do not change it to something else). The sampling clock is only used by the *chronogram* and will not show up in the timing diagram. It should not be connected to any other components and can be placed anywhere on main. Set the properties for *sysclk* to a 1 Tick high duration and a 1 Tick low duration (this is the default).
      2. Add a circuit master clock labeled *clk*. This is the clock that will be used to trigger all components in the circuit. Set the properties for *clk* to a 4 Tick high duration and a 4 Tick low duration.
      3. Set Simulate -> Tick Frequency to 4 Hertz. This will simulate a clock that ticks once per second, as in Figure 1.8. While the

actual tick frequency can be changed later to “speed up” the cir- cuit, a one-second tick is useful for learning how the *chronogram* works.

* + - 1. Click Simulate -> Chronogram to set up the *chronogram*. Fig- ure [1.9](#_bookmark108) illustrates the initial setup screen for the *chronogram*.

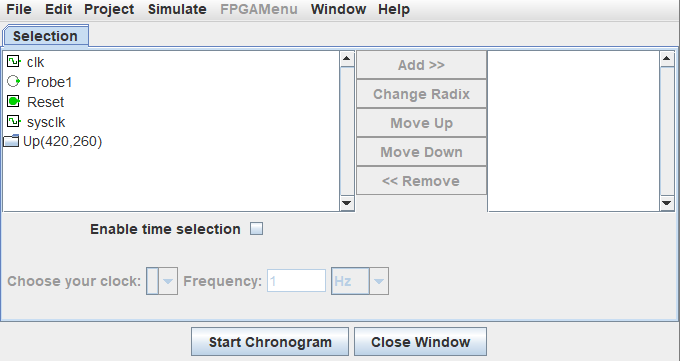


Figure 1.9: Set Up Chronogram

* + - 1. Click *sysclk* in the left panel and then click *Add »* to add that signal to the *chronogram*. The “-2” following the *sysclk* name in

the right panel indicates that it is a binary signal. It is probably *NOTE: sysclk must*

best to add the *sysclk* signal first so it is not overlooked.

* + - 1. Click *clk* in the left panel and then click *Add »* to add that signal to the *chronogram*.
      2. Click *Probe1* in the left panel and then click *Add »* to add that signal to the *chronogram*.
      3. Click “Enable time selection” and chose *clk* as the clock with a frequency of 1 Hertz.
      4. The *chronogram* setup should look like Figure 1.10.

*be added to the chronogram or it will not sample the circuit; however, the sysclk signal will not actually show up in the timing diagram.*

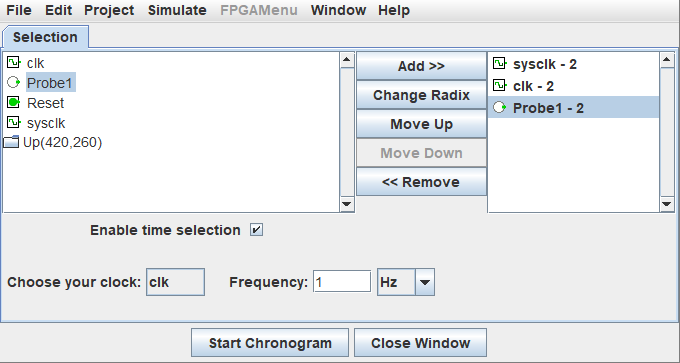


Figure 1.10: Chronogram Ready

* + - 1. Click *Start Chronogram* and the screen illustrated in Figure 1.11

pops up.

A screenshot of a computer

Description automatically generated

Figure 1.11: Chronogram Starting

* + - 1. Right-click on the *Probe1* signal and set the format for binary. The format can be set for any radix but to match this lab binary numbers should be specified.
      2. Right-click on the *Probe1* signal and enable *Expand* to see all four signals that create *Probe1*.
      3. At this point, the chronogram should look like Figure 1.11.

A screenshot of a computer

Description automatically generated

Figure 1.12: Chronogram At Zero Time

* + - 1. The *chronogram* has five buttons that control the simulator.

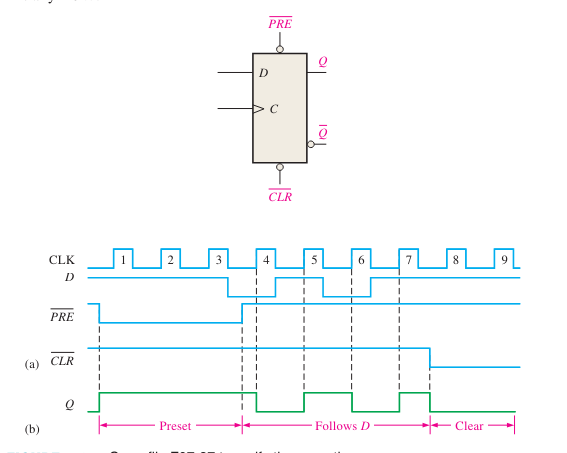


Figure 1.13: Chronogram Controls

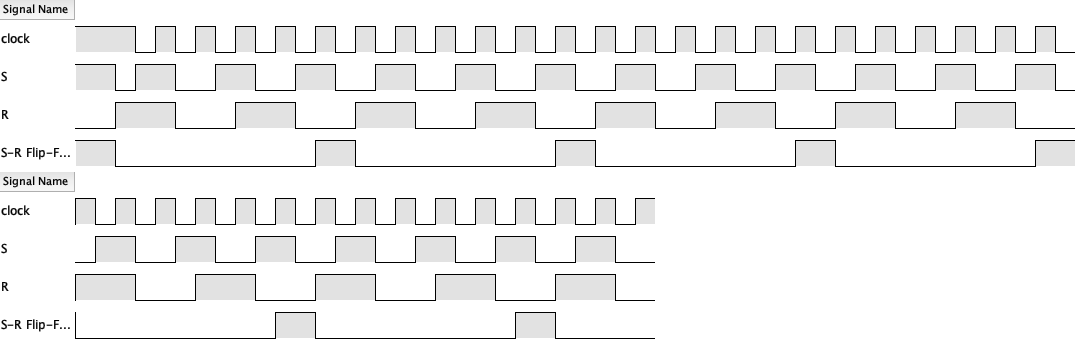
* Button One: Start/Stop the simulation.
* Button Two: Simulate one step.
* Button Three: Start/Stop *sysclk*. This will “turn on” the chronogram and begin creating a timing diagram.
* Button Four: Step one *sysclk* tick. This will tick the *sysclk* one time. Since this lab set up the *sysclk* for four ticks per second this button would need to be clicked four times to extend the timing diagram one second.
* Button Five: Step one *clk* tick. This extends the timing di- agram by one complete clock tick, or one second in this circuit.
  + - 1. Click button three to start the *chronogram* and watch the timing diagram unfold. After a few seconds click that button a second time to stop the *chronogram*.
      2. The following can be done once the timing diagram is complete.
* Click on the timing diagram to set the cursor (indicated by a red line). Once the cursor is set the values for each signal at the cursor’s location are printed next to the signal’s label on the left edge of the timing diagram.
* Hover the mouse over the timing diagram and roll the mouse wheel to zoom the timing diagram appearance.
  + Click “Export” to save the timing diagram signal levels in a text file. That file can later be loaded to reevaluate the timing diagram.
  + Click “Export as image” to save the timing diagram as a PNG file.

# Tasks for the Lab

1. Draw truth tables and timing diagram which have all the states of all types of flip-flops (D, T, JK, SR) kindly note in timing diagram do label states at each clock iterations e.g.



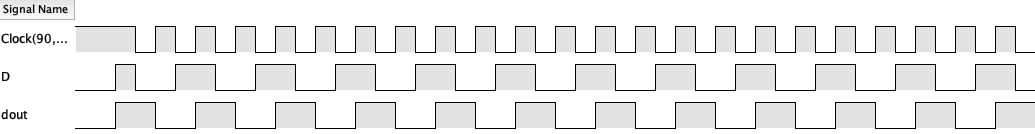
|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **R** | **Q** | **Q’** |
| 0 | 0 | No change | No change |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | undefined | undefined |



|  |  |  |  |
| --- | --- | --- | --- |
| D | CLOCK | Q | Q’ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

A white rectangular pattern with black squares

Description automatically generated with medium confidence



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **clock** | **J** | **K** | **Q** | **Q’** |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

A white and grey rectangular shapes

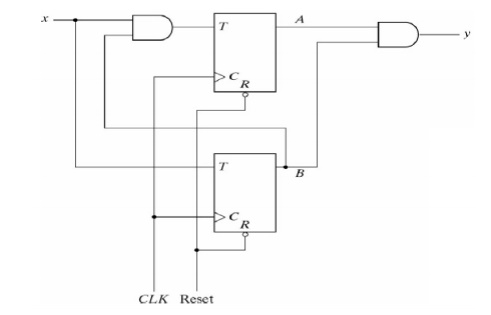
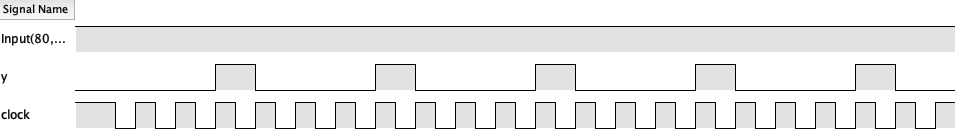
Description automatically generated with medium confidence

|  |  |  |
| --- | --- | --- |
| **T** | **Q** | **Q (t+1)** |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

A white and grey rectangular shapes

Description automatically generated with medium confidence

2. For the circuit below provide its truth table and Timing diagram for all values of x and reset.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | A current | B current | A next | B next | y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |